

MICROBASIC

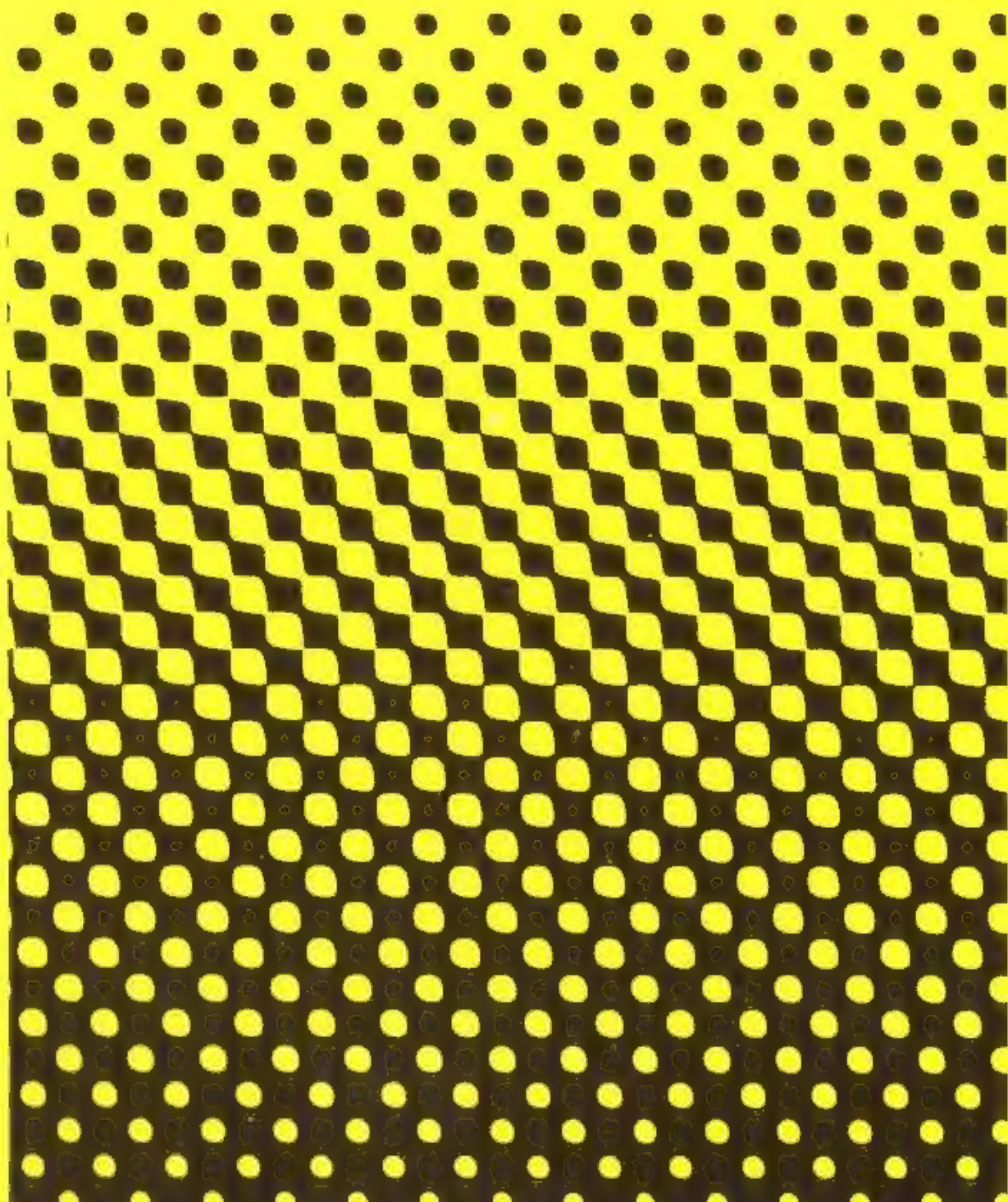
MICROBASIC

Basic SINCLAIR

Rafael PRADES

Rafael PRADES

BASIC SINCLAIR





Cuando en Junio de 1984 se empezó a gestar la creación del semanario MICRO-HOBBY, Rafael Prades fue la primera persona en contactar con la dirección del proyecto.

Nacido en agosto de 1956, casado y con tres hijos, ha realizado estudios de Maestría Industrial en la especialidad de Electrónica.

En 1974 entra en Standard Eléctrica, donde actualmente trabaja en el desarrollo del Sistema Doce de telefonía Digital como técnico de Hardware.

Curiosamente, y pese a ser un gran especialista en Hardware, se ha adaptado perfectamente a los temas de Software que le han sido encomendados, entre ellos, el Curso de Basic que nos ocupa.

El carácter eminentemente didáctico que imprime a todos sus artículos y, particularmente a este libro, se debe a su dilatada experiencia como Profesor de Fotografía, una de sus grandes pasiones, junto con el Montañismo.

HOBBY PRESS, S.A.
Editamos para gente inquieta.

MICROHOBBY

SEMANAL

BASIC SINCLAIR

Rafael PRADES

Con la aparición del primer número de MICROHOBBY comienza también una serie de capítulos dedicados al estudio del sistema BASIC como lenguaje de programación del ordenador Spectrum. El estudio, que será bastante completo, no se limitará sólo a definir las funciones de los comandos o sentencias, tratará al mismo tiempo de explicar todos los posibles argumentos asociados a éstos. Asimismo, cada tema irá acompañado de una larga y clara lista de ejemplos. El curso está pensado tanto para quienes se han comprado un ordenador Spectrum para pasar el rato y matar maripositos, como para aquellas personas que, teniendo cierta experiencia, deseen ampliar sus conocimientos sobre el tema. Una vez terminado el curso, presentaremos una serie de programas que serán explicados, analizados y convenientemente acompañados de diagramas de bloques y flujos.

HOP EDITA
HOBBY
PRESS, S.A.

MICROBASIC 1

BASIC SINCLAIR

Dos personas que quieran comunicarse necesitan un medio de expresión; éste puede ser el lenguaje hablado, escrito, la mímica, dibujos, etc...; de la misma manera, nosotros para poder comunicarnos con nuestro ordenador necesitamos conocer un lenguaje que él entienda. El utilizado por el Spectrum se denomina Lenguaje de Programación BASIC (iniciales en inglés de Beginner's All purpose Symbolic Instruction Code, que traducido al español significa «Código de instrucciones simbólicas de uso general para principiantes»).

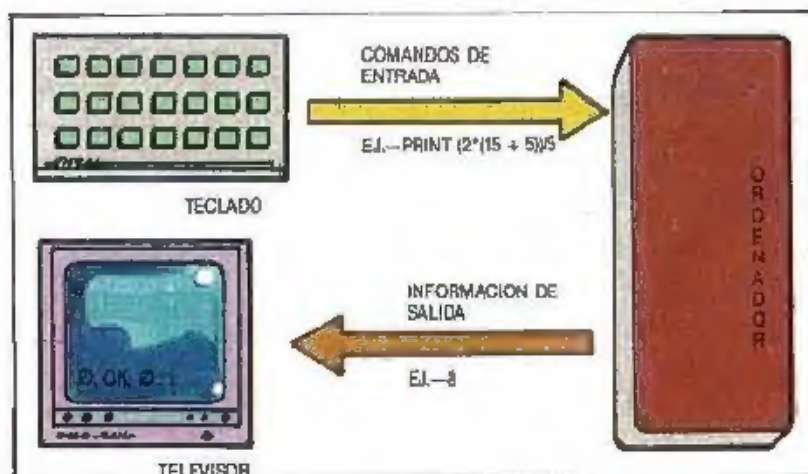
El lenguaje del Spectrum

El BASIC es un lenguaje utilizado también por otros ordenadores personales, aunque cada cual tiene ciertas particularidades que lo diferencian de los demás. En esta serie de artículos sólo vamos a tratar el lenguaje BASIC que el Sinclair Spectrum entiende.

Ya sabemos cuál es el medio de expresión que debemos utilizar con nuestro ordenador, pero ¿cómo se emplea? La respuesta es sencilla: el BASIC consta de una serie de comandos o sentencias que el programador introduce en el ordenador mediante el manejo del teclado. El aparato nos devuelve la información pedida en la pantalla del televisor. Por ejemplo, si quisiéramos que el ordenador nos calculara la operación:

$$\frac{2(15 + 5)}{5}$$

2 MICROBASIC



El ordenador, una vez recibido el comando que se ha introducido mediante el teclado, realiza las operaciones necesarias y visualiza el resultado en la pantalla del televisor.

y que nos devolviera el resultado, introduciríamos en el teclado el comando:

`PRINT 2 * (15 + 5) / 5`

El ordenador, una vez recibido el comando, calcula la expresión aritmética y posteriormente visualiza en la pantalla del televisor el resultado.

Como decíamos más arriba, el lenguaje BASIC consta de una serie de comandos o sentencias con las que se establece un diálogo o comunicación con el ordenador, pero ¿qué son las sentencias? Los comandos o sentencias son palabras clave derivadas del idioma inglés (PRINT, RUN, STOP...); estos términos, al ser introducidos en el ordenador mediante el teclado, indican a éste qué función debe de realizar.

Normalmente, las sentencias por sí solas no podrían realizar sus funciones si no

fueran acompañadas de un argumento, ya que éste aporta el dato o datos necesarios para que el comando o sentencia pueda ser ejecutado. Los argumentos pueden ser números, variables, expresiones aritméticas o cadenas de caracteres.

Se denomina *instrucción* al conjunto de una sentencia con su correspondiente argumento, y tiene la siguiente estructura:

INSTRUCCIÓN	
SENTENCIA	ARGUMENTO

Por ejemplo:

- a) El argumento es un número.
Función: saltar a la subrutina localizada en la línea 350.

SENTENCIA	ARGUMENTO
GOSUB	350

- b) El argumento es una variable.
Función: Leer un dato de una tabla y asignarlo a la variable «C».

SENTENCIA	ARGUMENTO
READ	C

- c) El argumento es una expresión matemática.
Función: asignar a la variable «X» el resultado de la operación:

$$(15 \times 2) + (8/3) \\ 5$$

SENTENCIA	ARGUMENTO
LET	$X = ((15 * 2) + (8/3)) / 5$

- d) Y, por último, el argumento es una cadena de caracteres.
Función: visualizar la siguiente cadena alfanumérica «Revista Microhobby».

SENTENCIA	ARGUMENTO
PRINT	«Revista Microhobby»

No se preocupe si aún no entiende qué es una subrutina, qué es una variable o cómo se introducen estas sentencias, ya que esto se irá viendo en capítulos siguientes.

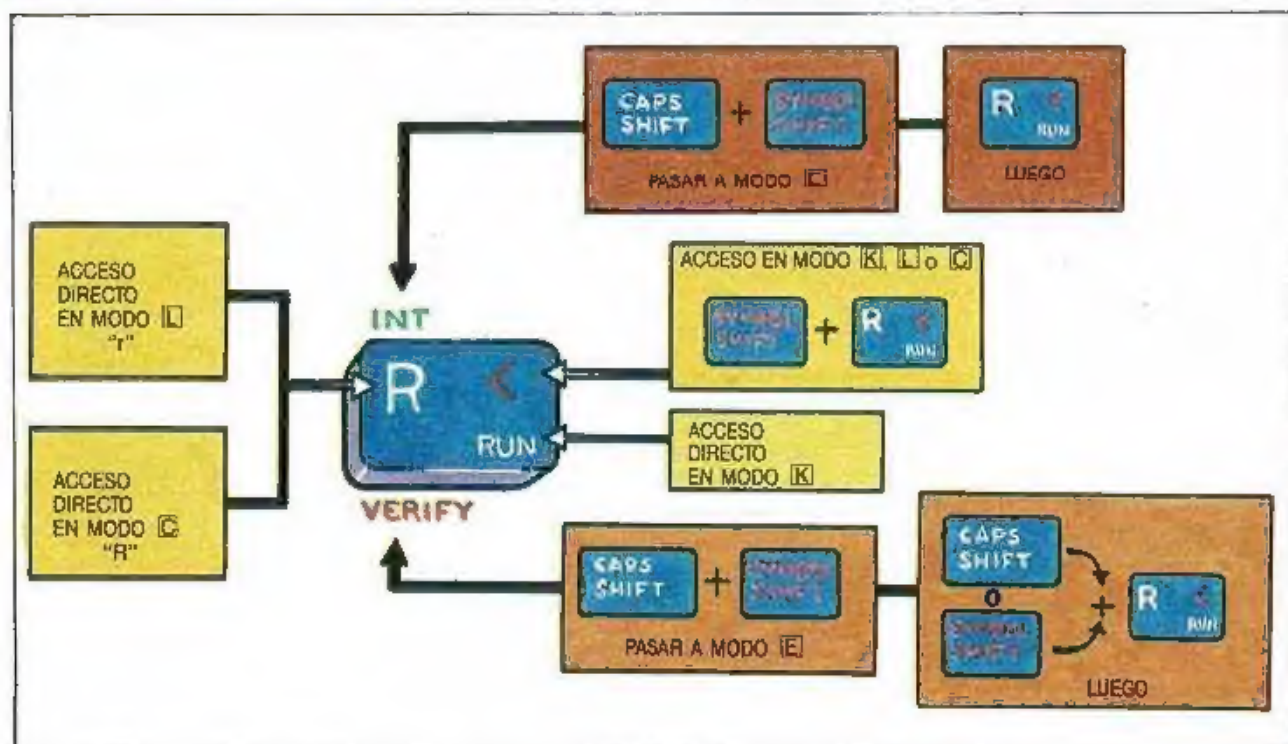
Acceso al teclado

Una de las características del Spectrum consiste en que no es necesario teclear los comandos o sentencias letra a letra, sino que éstos ya están definidos y asignados a una tecla. Por ejemplo: para acceder a la sentencia PRINT no se necesita pulsar las teclas correspondientes a las letras P, R, I, N y T; con sólo accionar la «P» aparece escrita en la pantalla del televisor el comando PRINT.

El Spectrum tiene 40 teclas; con éstas sería imposible albergar los 88 diferentes comandos que existen, así como las diferentes funciones de control, los distintos gráficos

definidos, etc...; esto obliga a que las teclas tengan funciones múltiples. La manera de acceder a las diferentes funciones que posea una tecla viene determinada por el color y la posición que ésta ocupa dentro o fuera de la tecla; también depende del modo en que se encuentre el cursor parpadeante que aparece en la parte inferior de la pantalla.

MODOS **K**.—De entrada, al conectar el ordenador, según indica en el capítulo primero del manual de introducción que acompaña a su Spectrum, aparece el mensaje «© 1982 Sinclair Research Ltd»; este mensaje desaparece al presionar la tecla ENTER y en su lugar aparece el cursor parpadeante indicando el modo **K**; en este modo se tiene acceso a los números situados en la fila de teclas superior o a las sentencias pintadas de blanco, dentro de las tres filas de teclas restantes.



En el Spectrum, las teclas son multifunción. Cada una de ellas tiene diversas utilidades, por lo que es imprescindible un correcto manejo de los «modos».

ZX Spectrum



Tomemos, por ejemplo, la tecla correspondiente a la letra R; en modo **R** tendríamos acceso a la sentencia RUN.

MODO L.—Una vez visualizada la palabra RUN en la pantalla, el cursor cambia a modo **L**; con lo que se tiene acceso a la letra dibujada en blanco dentro de la tecla, o pulsando simultáneamente la tecla SYMBOL SHIFT y la tecla correspondiente tendríamos acceso a la sentencia o signo dibujado en rojo, dentro de la tecla, en la parte superior derecha. Tomando el mismo ejemplo y trabajando en modo **L** la tecla R, aparecería en la pantalla la letra r minúscula; si quisiéramos que apareciera como R mayúscula tendríamos que utilizar simultáneamente, como en una máquina de escribir, CAPS SHIFT y la tecla R. Pulsando simultáneamente SYMBOL SHIFT y la tecla R se visualizaría en la pantalla el signo < (menor que).

MODO C.—El modo **C** es una variante del **L**, ya que permite escribir siempre en mayúsculas sin tener que estar pulsando continuamente la tecla CAPS SHIFT; para acceder a este modo basta con pulsar simultáneamente CAPS SHIFT y CAPS LOCK, situado en la tecla correspondiente al número 2. A partir de este momento, el modo **C** sustituye al modo **L**. Presione la tecla R y comprobará que la R mayúscula aparecerá en lugar de la r minúscula.

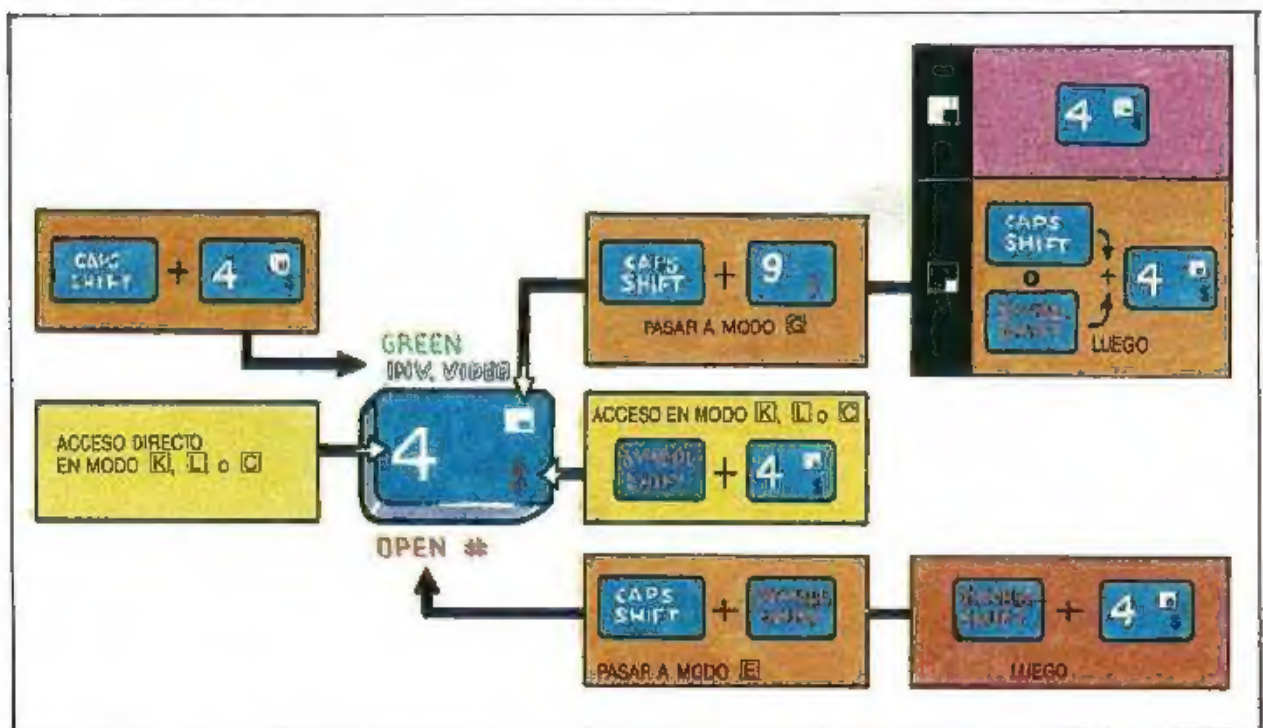
Si desea volver otra vez al modo **L**, sólo hay que repetir de nuevo la secuencia CAPS SHIFT + CAPS LOCK y de nuevo aparecerá el cursor en modo **L**. El modo que siempre presenta el Spectrum, si no se le indica lo contrario, es el **L** (minúsculas).

MODO E.—Para poder acceder a las sentencias situa-

das en la parte exterior de la tecla, en color verde las de la parte superior y en color rojo las de la inferior, es necesario pasar al modo **E**.

Al modo **E** se accede pulsando simultáneamente las teclas CAPS SHIFT y SYMBOL SHIFT, en este momento el cursor cambia a modo **E**, una vez realizada esta operación tenemos dos opciones: o bien acceder a las sentencias pintadas en verde, o bien a las pintadas en rojo. Para seleccionar las primeras basta con usar la tecla correspondiente una vez que estemos en modo **E**; para acceder a las segundas es necesario, además de estar en **E**, pulsar simultáneamente SYMBOL SHIFT o CAPS SHIFT y la tecla correspondiente.

En el ejemplo anterior, si se pasa a modo **E** y se pulsa la tecla R, se visualizaría en la pantalla de su televisor o mo-



Las teclas de la fila superior tienen un tratamiento distinto, ya que contienen los gráficos predefinidos y una serie de funciones de control.

6 MICROBASIC

nitor la sentencia INT; pero si, estando en modo **E**, se accionan simultáneamente las teclas SYMBOL SHIFT o CAPS SHIFT y R, aparecerá en la pantalla el comando VERIFY. En ambos casos se observará que después de visualizarse la sentencia correspondiente (INT o VERIFY), el cursor cambia a modo **L** o **C**.

Si, por error, al introducir una sentencia que no corresponde al modo **E** se ha pasado a éste, para volver al modo anterior bastaría con repetir la secuencia CAPS SHIFT + SYMBOL SHIFT.

Las teclas de la fila superior tienen un tratamiento distinto, ya que contienen los gráficos predefinidos en el Spectrum y una serie de funciones de control...

Las funciones de control se encuentran fuera de las teclas pintadas de blanco y en la parte superior. Estas funciones, como por ejemplo EDIT, DELETE, desplazamientos del cursor, etc..., ya irán siendo tratadas en los siguientes párrafos; ahora lo más importante es conocer como acceder a ellas; para ello basta con pulsar simultáneamente las teclas CAPS SHIFT y la tecla correspondiente a la función deseada; es decir, si quisiéramos acceder a la función EDIT bastaría con pulsar, a la vez las teclas CAPS SHIFT y la correspondiente al número 1.

MODO **G.**—Para acceder a los gráficos situados en las teclas con los números 1 a 8 y a los definidos por el usuario, situados en las teclas con las letras de la A a la U, hay que pasar al modo **G**. Para

acceder a **G** es necesario hacer uso de la función GRAPHICS; que como ya conocen, se accede oprimiendo simultáneamente la tecla CAPS SHIFT y la que lleva el número 9. Una vez que la operación anterior ha sido realizada, el cursor parpadeante pasa a modo **G**.

Los símbolos predefinidos están representados en la esquina superior derecha de las teclas 1 a 8; para visualizar cualquiera de estos símbolos en la pantalla sólo es necesario pulsar la tecla correspondiente al gráfico elegido. También se pueden visualizar estos gráficos en negativo; es decir, que las partes claras se conviertan en oscuras y las oscuras en claras; para ello basta con accionar, a la vez que se selecciona un gráfico, la tecla CAPS SHIFT o SYMBOL SHIFT.

Para visualizar los gráficos definidos por el usuario es necesario que estos estén ya definidos con anterioridad; si algún lector no sabe todavía cómo definir sus propios gráficos, no se preocupe; en otro artículo de este curso se tratará este tema, y en otras secciones de «Microhobby» también se comentará. Si los gráficos ya estuvieran definidos, bastaría para visualizarlos con pulsar la tecla correspondiente a la que se le asignó; es decir, si definimos un gráfico y lo asignamos a la tecla con la letra R, para visualizarlo habría que pasar a modo **G** y pulsar simplemente la tecla R.

Aquellas letras entre la A y la U que no tengan un gráfico asignado, al pasar a modo **G** y seleccionarias, se visualizaría en la pantalla la letra correspondiente a la letra pulsa-

da, escrita en mayúsculas, aunque el modo elegido anteriormente fuera el **L**.

Para poder volver al modo anterior basta con pulsar la tecla GRAPHICS, situada en el número 9, y el cursor volverá a parpadear en el modo anterior. Desde el modo **G** también se puede pasar directamente al modo **E**, pulsando simultáneamente CAPS SHIFT y SYMBOL SHIFT.

Para acceder a los signos en rojo, situados en la esquina inferior derecha, basta con pulsar simultáneamente SYMBOL SHIFT y la tecla correspondiente.

A los números del ϕ al 9 se puede acceder tanto en modo **K** como en **L** o **C**.

Las sentencias pintadas en rojo y situadas en la parte inferior de las teclas no tienen el mismo tratamiento que sus homólogas de las restantes filas de teclas, ya que para acceder a éstas sólo es posible hacerlo cuando estemos en modo **E**, pulsando SYMBOL SHIFT más la tecla correspondiente; sin embargo, en las restantes filas de teclas podría ser o bien con CAPS SHIFT o con SYMBOL SHIFT.

Las funciones de color, situadas en la parte exterior de las teclas ϕ a 7, indican la correspondencia que hay entre colores y números, ya que cuando en algún comando haya que indicar una función de color, ésta será dada por su número; es decir, si en una sentencia de visualización de pantalla hay que indicar que ésta sea en color rojo (RED), habrá que seleccionar la tecla con el número 2.

Esta guía facilita la búsqueda de sentencias, funciones o símbolos, cuando aún no se tiene la suficiente destreza de manejo o cuando se produce alguna duda. Las sentencias están ordenadas por orden alfabético; para una mejor localización, a la izquierda de la sentencia aparece la inicial de ésta, y, a la derecha, la tecla donde está situada.

A	ABS	G
	ACS	W
	AND	Y
	ASN	O
	AT	I
	ATN	E
	ATTR	L

M	MERGE	T
	MOVE	S

N	NEW	A
	NEXT	N
	NOT	S

B	BEEP	Z
	BIN	E
	BORDER	B
	BRIGHT	B

O	OPEN #	4
	OR	U
	OUT	O
	OVER	N

C	CAT	S
	CHR\$	U
	CIRCLE	H
	CLEAR	X
	CLOSE #	5
	CLS	V
	CODE	I
	CONT	C
	COPY	Z
	COS	W

P	PAPER	C
	PAUSE	M
	PEEK	O
	PI	M
	PLOT	Q
	POINT	B
	POKE	O
	PRINT	P

R	RAND	T
	READ	A
	REM	E
	RESTORE	S
	RETURN	Y
	RND	T
	RUN	R

D	DATA	D
	DEF FN	1
	DIM	D
	DRAW	W

S	SAVE	S
	SCREEN \$	K
	SGN	F
	SIN	Q
	SOR	H
	STEP	D
	STOP	A
	STR\$	Y

E	ERASE	7
	EXP	X

F	FLASH	V
	FN	2
	FOR	F
	FORMAT	Q

T	TAB	P
	TAN	E
	THEN	G
	TO	R

G	GOSUB	H
	GOTO	G

U	USR	L
---	-----	---

I	IF	U
	IN	I
	INK	X
	INKEY\$	N
	INPUT	I
	INT	R
	INVERSE	M

















V	VAL	J
	VAL\$	J
	VERIFY	R

L	LEN	K
	LET	L
	LINE	3
	LIST	K
	LLIST	V
	LN	2
	LOAD	J
	L PRINT	C

FUNCIONES DE CONTROL	
BREAK	SPACE
CAPS LOCK	2
DELETE	0
EDIT	1
GRAPHICS	9
INV. VIDEO	4
TRUE VIDEO	3

OTRAS FUNCIONES	
CAPS SHIFT	IZDA.Z
ENTER	DCHA.L
SYMBOL SHIFT	DCHA.M

DESPLAZAMIENTOS DEL CURSOR	
↑	7
↓	6
→	8
←	5

GRAFICOS		
		1
		2
		3
		4
		5
		6
		7
		8

COLORES	
BLACK	0
BLUE	1
CYAN	5
GREEN	4
MAGENTA	3
RED	2
WHITE	7
YELLOW	6

SIMBOLOS	
@	2
!	P
#	3
\$	4
%	X
&	6
'	A
(8
)	9
*	Y
+	U
,	F
-	G
.	D
/	V
:	S
;	0
<	1
=	C
>	M
[N
\	O
]	Z
^	P
_	7
`	K
{	J
	B
}	L
~	H
~	5
>	T
<	R
<>	W
>=	E
<=	Q

(Doblar por aquí)

Edición de programas

En el momento de introducir una instrucción en el ordenador se nos presentan dos opciones: que se ejecute nada más ser introducida o que quede almacenada en la memoria del ordenador para su posterior ejecución.

En la primera opción, una vez ejecutada la instrucción, si se desea repetir su ejecución es necesario teclear de nuevo; en el segundo caso no es necesario, ya que la primera vez que se introdujo quedó almacenada en la memoria del ordenador y podemos ejecutarla tantas veces como queramos, siempre y cuando no desconectemos la clavija de alimentación de 9V DC.

Para introducir una *instrucción* de ejecución inmediata, sólo es necesario teclearla y pulsar la tecla ENTER, que indica el ordenador que la *instrucción* se ha terminado de teclear; en ese momento el ordenador la analiza para que no tenga ningún error de sintaxis; es decir, que el argumento está correctamente tecleado y que está en concordia con el tipo de sentencia. Si todo ha sido correcto, la *instrucción* se ejecuta inmediatamente.

Para introducir una *instrucción* que no se ejecute inmediatamente, es necesario asignarle un número de línea comprendido entre 1 y 9999; este número se teclea estando el cursor en modo **K**; a continuación se teclea la *instrucción* y posteriormente, como en el caso anterior, se pulsa la tecla ENTER. Se analizan los posibles errores que la *instrucción* podría contener y si todo es correcto, no se ejecuta, sino que pasa a memoria; esto se puede comprobar, ya que

se visualiza en la parte superior de la pantalla.

Para poder ejecutar esa *instrucción*, es necesario ejecutar con anterioridad la sentencia inmediata RUN; para ello basta con pulsar las teclas R y ENTER. Como hemos dicho anteriormente, esa *instrucción* se puede repetir, tantas veces como queramos, simplemente introduciendo la sentencia RUN y pulsando ENTER.

Veamos unos ejemplos; pruebe a introducir la siguiente *instrucción* inmediata:

PRINT "Curso BASIC/Spectrum"

al terminar de teclear y pulsar ENTER se ejecutará la *instrucción* y aparecerá en la parte superior de la pantalla, la cadena alfanumérica: Curso BASIC/Spectrum, y en la parte inferior un mensaje que envía el ordenador, indicando que la *instrucción* ha sido ejecutada correctamente, "OK, 1". Si quiere repetir la ejecución, es necesario repetir también la *instrucción*.

Para que ésta no se ejecute hasta que usted quiera, debe de asignarle un número de línea, por ejemplo el 10; por tanto teclee:

10 PRINT "Curso BASIC/Spectrum"

al pulsar ENTER, comprobará que la *instrucción* tecleada pasa a la parte superior de la pantalla; también observará que después del número de línea 10, el ordenador ha colocado el símbolo >; éste es un indicador de presencia, conocido también como "prompt" que indica cuál es la última línea editada.

Para poder ejecutar esa *instrucción*, deberá introducir

RUN, y como siempre pulsar ENTER. El listado de la *instrucción* se borrará, y ésta se ejecutará de la misma manera que si hubiese sido introducida directamente. Aunque el listado haya desaparecido de la pantalla, no se preocupe: está almacenado en la memoria del ordenador para poder visualizarlo de nuevo, simplemente pulse la tecla ENTER. Podrá volver a ejecutar la *instrucción*, sin tener que teclearla otra vez, introduciendo la sentencia RUN y pulsando ENTER.

El numerar una sola *instrucción*, en principio, no tiene mucha utilidad, lo que sí tiene y bastante, es numerar una cantidad más o menos larga de *instrucciones*; esto es lo que se conoce como *programa*. Un programa es, por tanto, una secuencia de *instrucciones* ordenadas que realizan una función determinada.

Un ejemplo de estructura de un programa puede ser el siguiente:


10	Instrucción	n.º 1
20	Instrucción	n.º 2
30	Instrucción	n.º 3
100	Instrucción	n.º 10

En el lenguaje BASIC, las *instrucciones* se suelen numerar de diez en diez; esto se debe a que muy pocas veces un programa funciona a la primera; siempre es necesario modificar, borrar o añadir alguna *instrucción* que se nos quedó olvidada en el tintero. Si las tenemos numeradas a intervalos de diez, siempre tenemos la posibilidad de insertar una *instrucción*, asignándole un número de línea intermedio, ya que el ordenador, cuando eje-

cuta el programa que tiene en memoria, siempre lo hace empezando por la instrucción con el número de línea más bajo y continúa en orden creciente.

En el ejemplo anterior, si queremos insertar una instrucción entre las líneas 10 y 20, le asignaríamos, por ejemplo, el número 15, de manera que el programa quedaría de la siguiente forma.

10	Instrucción	n.º 1
15	instrucción	n.º 2
20	instrucción	n.º 3
30	instrucción	n.º 4
100	instrucción	n.º 11

En el Spectrum, los números de línea tienen que estar comprendidos entre el 1 y el 9999; cualquier instrucción que se asigne con una numeración distinta, será rechazada por el ordenador. En el caso de que se asigne a una instrucción un número de línea igual a 0, la pantalla se limpiará de caracteres, y en la zona reservada para los informes del ordenador, aparecerá el mensaje: C Nonsense in BASIC, 0 : 1. Para volver a recuperar el listado, si lo hubiera, pulse ENTER. Si el número de línea fuese superior a 9999 al pulsar ENTER aparecerá una  parpadeante a la derecha del número de línea; en el capítulo denominado "Corrección de errores", se dan orientaciones necesarias para poder corregir este fallo.

Se habrá dado cuenta que, según se van introduciendo las instrucciones, el prompt > se va desplazando, y siempre apunta a la última línea editada.

Dentro de una misma línea se pueden introducir varias

10 MICROBASIC



Ejemplo de edición de un programa.

```

10 REM *****
   *          *
   *  CURSO BASIC  *
   *          *
   *   EJER - 1    *
   *          *
   * *****      *

50 BORDER 1: PAPER 7: INK 2  C
LS
60 GO SUB 1000
90 REM *****
   *          *
   *  REJILLA  *
   *          *
   * *****

100 FOR n=8 TO 248 STEP 8
110 PLOT n,175
120 DRAW 0,-175
130 BEEP 0.05,n/8
140 NEXT n
150 FOR n=167 TO 7 STEP -8
160 PLOT 0,n
170 DRAW 255,0
180 BEEP 0.05,n/8
190 NEXT n
200 REM

```

```
*****
* MENSAJE VERTICAL *
*****
```

```
210 RESTORE 270
220 FOR n=1 TO 88
230 READ y, READ x
240 PRINT AT y,x;CHR$ 20+CHR$ 1
,CHR$ 32
250 BEEP 0.05,y
260 NEXT n
270 DATA 3,2,4,2,5,2,6,2,7,2,3,
5,7,5
280 DATA 2,8,3,8,4,8,5,8,6,8,7,
8,2,11,3,11,4,11,5,11,6,11,7,11
290 DATA 2,14,3,14,4,14,5,14,6,
14,7,14,8,14,6,16,7,17,3,18,4,18
,8,18
300 DATA 3,21,4,21,7,21,3,24,6,
24,7,24
310 DATA 3,27,4,27,5,27,6,27,7,
27,3,30,4,30,6,30,6,30,7,30
320 DATA 11,3,12,3,13,3,14,3,15
,3,16,3,17,3,12,6,13,6,15,6,16,6
330 DATA 12,9,13,9,14,9,15,9,16
,9,17,9,12,12,13,12,14,12,15,12,
16,12,17,12
340 DATA 12,15,13,15,16,15,12,1
8,15,16,16,16
350 DATA 12,22,13,22,14,22,15,2
2,16,22
360 DATA 12,26,13,26,14,26,15,2
6,16,26,12,29,16,29
400 REM
```

instrucciones; éstas tienen que estar separadas por el signo . Utilizando el separador, el programa ocupa menos memoria y la ejecución de éste se hace más rápida; sin embargo, tiene la desventaja de que a veces los listados de los programas no quedan lo suficientemente claros, como para poder interpretarlos rápidamente.

Un ejemplo de utilización del separador, podría ser el siguiente:

```
1Ø Instrucción n.º
2Ø Instrucción n.º 2 3 4
3Ø Instrucción n.º 5
1ØØ Instrucción n.º 12
```

Corrección de errores

Mientras efectúa la laboriosa tarea de edición de un programa, seguro que alguna vez se habrá olvidado de introducir alguna instrucción, un número de línea que no correspondía o bien se habrá encontrado que en el momento de insertarla en memoria no se podía, ya que aparecía una **?** parpadeante, indicando un error de sintaxis. En este capítulo, se va a tratar de aclarar las posibles dudas que se tengan sobre borrado, modificación o inserción de nuevas líneas.

Para hablar de las posibles correcciones a efectuar en

nuestro programa, vamos a distinguir si las instrucciones están insertadas en memoria o no, ya que su tratamiento depende de este detalle.

Para modificar una instrucción que estamos tecleando, es necesario hacer uso de varias de las funciones situadas en la fila superior de teclas, éstas son: los desplazamientos del cursor, izquierda (**←**) y derecha (**→**), y la función DELETE. Los desplazamientos del cursor no necesitan casi explicación, ya que como su propio nombre indica, desplazan el cursor parpadeante en la dirección que indica la flecha, y la función DELETE borra el carácter situado a la izquierda del cursor. Estas funciones se seleccionan apretando la tecla CAPS SHIFT simultáneamente con la de función correspondiente; al igual que ocurre con cualquier otra tecla, si se mantienen apretadas ambas durante cierto tiempo, la función se repite. En el modo **G**, para seleccionar la función DELETE no se necesita pulsar CAPS SHIFT; simplemente apretando la tecla con el número **Ø** se borra el carácter anterior al cursor.

Una vez explicadas las funciones, para poder realizar una modificación, es necesario desplazar el cursor a la izquierda, hasta situarlo en el lugar que queramos; ya en él podemos, o bien insertar algún carácter que se nos olvidó o bien hacer uso de la función DELETE. Si quisiéramos seguir escribiendo al final de la línea será necesario ir desplazando el cursor hacia la derecha, hasta llegar a él.

En caso de que apareciera una **?** parpadeante al inten-

tar insertar una instrucción, la secuencia de operación sería a misma; la interrogación nos facilita a búsqueda del error, ya que se coloca al lado de él.

Para borrar una línea completa que estemos tecleando podríamos, haciendo uso de la función DELETE, ir borrando carácter por carácter pero cuando la línea es larga resulta más cómodo operar de otra manera: se accede a la función EDIT; para ello basta accionar simultáneamente la tecla CAPS SHIFT y la corres-

pondiente al número 1; en ese momento nos desaparece la línea que queríamos borrar, y en su lugar aparece una copia de la última línea editada, es decir, la que en el listado contenía el prompt >. Para reestablecer esta línea a la memoria, basta con pulsar ENTER.

Una vez que las sentencias están en memoria, para insertar una nueva simplemente basta, como ya dijimos en el capítulo "Edición de programas", asignarle una numeración intermedia; es decir, si queremos insertar dos nuevas

instrucciones entre las de líneas 110 y 120, podríamos asignarles los números de línea 113 y 117; de esta manera, el ordenador las ejecutará después de la 110 y antes de la 120.

Para borrar una instrucción, basta con teclear su número de línea que tenía antes; este método es cómodo cuando las instrucciones son pequeñas, pero cuando, por el contrario, son largas, es más cómodo hacer uso de la función EDIT. Cuando se hace uso de esta función, una copia de la línea



Aspecto del teclado profesional del ZX Spectrum +, incluyendo 18 nuevas teclas.

12 MICROBASIC

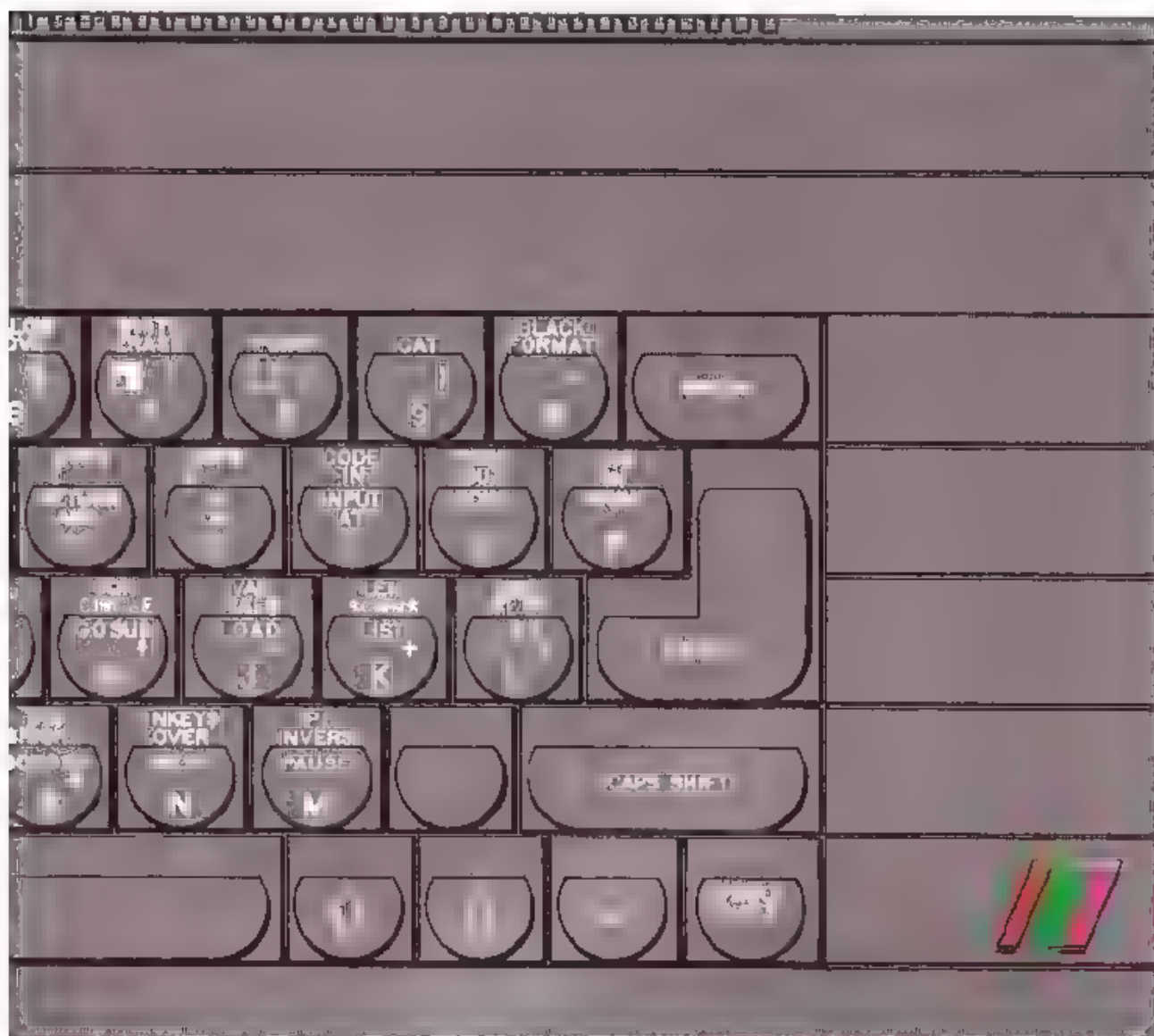
que contenga el prompt > aparece en la zona inferior de la pantalla; para poder desplazar el prompt hasta la instrucción que deseamos modificar, es necesario hacer uso de las funciones de desplazamiento del cursor, arriba (↑) y abajo (↓), situadas en las teclas con los números 7 y 6 respectivamente; estas funciones, al igual que EDIT, se realizan con ayuda de la tecla CAPS SHIFT.

Una vez que la línea a modificar está en la parte inferior, ésta se rectifica de la misma manera que cuando todavía

no está en memoria, es decir, con los desplazamientos de cursor derecha e izquierda y con la función DELETE; cuando la línea está corregida se pulsa ENTER. Si se usa la función EDIT para modificar un número de línea, al pulsar ENTER, no desaparece la instrucción con la numeración antigua, sino que se mantienen ambas en memoria, como se puede constatar por la información que se visualiza en pantalla. Esto puede servir para copiar una instrucción tantas veces como queramos,

con sólo darle una numeración de línea distinta; si, por el contrario, lo que deseábamos era cambiar el número de instrucción, tendremos que borrar la instrucción correspondiente a la numeración antigua; como hemos explicado anteriormente, esto se hace tecleando el número de línea y pulsando ENTER.

Existe otro método para modificar una línea sin tener que estar desplazando el prompt de línea en línea, este método lo explicaremos cuando veamos la sentencia LIST.



Ejercicio

Como comprobación de que ha entendido lo explicado hasta este momento, intente editar el programa que a continuación le proponemos, que combine los semigráficos con sonido y color.

No se preocupe si no entiende el significado de las sentencias o la filosofía del programa; ya que el objeto de éste es practicar el acceso al teclado, la edición y la corrección de los errores que pudieran surgir.

El programa que proponemos es un poco enigmático, hasta que no lo ejecute no sabrá el contenido del mensaje que aparece en pantalla. Aquellos lectores que entiendan algo de BASIC, y les guste lo misterioso, pueden intentar descifrar este mensaje, ya que la clave está en las sentencias DATA.

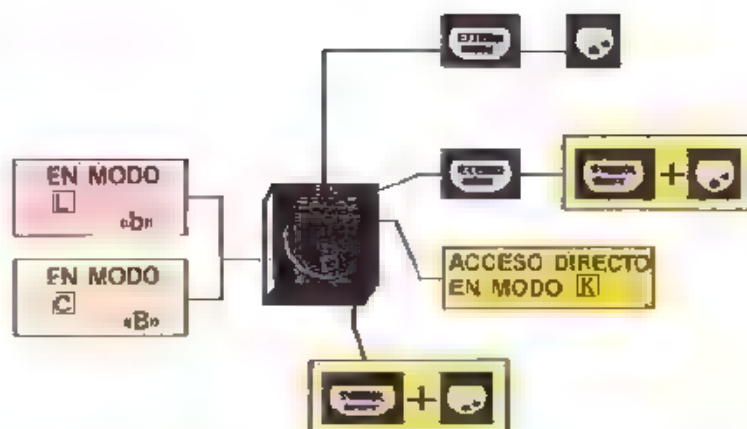
Es fácil de teclear, aunque hay que tener cuidado al pulsar los argumentos de las sentencias DATA, ya que si algún dato estuviese cambiado o faltase, el mensaje se vería desvirtuado. Cuando termine de teclear las 78 líneas de que se compone el programa, podrá ejecutarlo, pulsando RUN y ENTER.

Si todo es correcto, cuando termine de ejecutarse el programa, el ordenador enviará el siguiente mensaje:

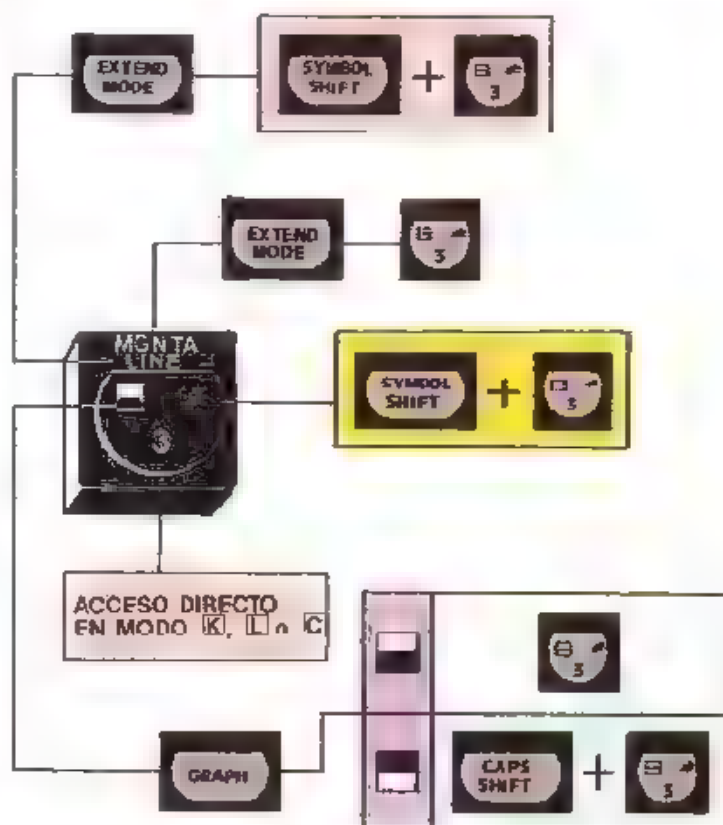
9 STOP statement, 999 : 1 si el ordenador enviara otro distinto, revise las instrucciones o los números de línea, ya que seguramente habrá cometido algún error. Si tiene dudas, no sienta reparo en volver a repasar los capítulos anteriores.

Esperamos que no tengan

14 MICROBASIC



Descripción del manejo de los modos en el nuevo teclado.



La fila superior tiene un tratamiento ligeramente distinto.

ningún tipo de dificultad y que el programa lo edite y ejecute correctamente.

Teclado del «ZX Spectrum +»

El «ZX Spectrum +», también conocido como «Spectrum Plus», presenta algunas diferencias con respecto a su

homólogo. Aparte de su aspecto exterior, que evidentemente es distinto, se percibe que existe una diferencia en cuanto al número de teclas, ya que éste tiene 58 en lugar de 40. En la tecla adjunta se observa cuáles son las nuevas teclas incorporadas y en la figura representativa del tecla-

do su localización. Las demás teclas, a excepción del SPACE, mantienen su situación y funcionamiento anterior. La tecla «SPACE» ha sido sustituida, al igual que en otros ordenadores, por una barra espaciadora semejante a la que incorporan las máquinas de escribir.

Otra de las diferencias del teclado, es la duplicidad de las teclas «CAPS SHIFT» y «SYMBOL SHIFT», esto proporciona un mejor manejo de estas funciones, ya que al estar situadas a ambos lados del teclado podrán ser utilizadas con cualquiera de las dos manos.

Modos **L** y **C**

Después de haber pulsado una tecla, el modo que presenta el «Spectrum +» por defecto es el **L** (Letter Mode), de este modo al pulsar una tecla, la letra inscrita en su interior se visualiza en minúscula; si desea que aparezca en mayúscula sin tener que cambiar de modo, es necesario pulsar simultáneamente junto con la letra elegida, cualquiera de las teclas «CAPS SHIFT». Cuando el texto a escribir en mayúscula es largo, conviene pasar al modo **C** (Capitals Mode). En el «Spectrum +» para acceder a este modo, basta simplemente con pulsar la tecla «CAPS LOCK»; a partir de ese momento, el modo **C** sustituye al **L** hasta que se pulse de nuevo esta tecla.

Modo **E**

El «Spectrum +» tiene una tecla específica para pasar a modo extendido **E**; ésta se denomina «EXTEND MODE». Para seleccionar la sentencia situada en la parte superior de a tecla que en el modelo an-

```
*****
*                               *
*  MENSAJE HORIZONTAL        *
*                               *
*****

410 RESTORE 470
420 FOR n=1 TO 45
430 READ y: READ x
440 PRINT AT y,x;CHR$ 20+CHR$ 1;CHR$ 32
450 BEEP 0.05,x
460 NEXT n
470 DATA 2,3,2,4,2,15,2,16,2,17,2,22,2,23,2,28,2,29
480 DATA 5,15,5,16,5,17,5,22,5,23
490 DATA 8,3,8,4,8,9,8,10,8,22,8,23,8,28,8,29
500 DATA 11,4,11,5,11,10,11,11,11,16,11,17,11,21,11,22,11,23,11,27,11,28
510 DATA 14,4,14,5,14,10,14,11,14,16,14,17
520 DATA 17,4,17,5,17,16,17,17,17,21,17,22,17,23,17,27,17,28
600 REM

*****
*                               *
*  TRIANGON                   *
*                               *
*****

610 FOR y=0 TO 21
620 FOR x=0 TO 31
640 IF (x AND y)=0 THEN GO TO 660
650 IF SCREEN$ (y,x)=CHR$ 32 THEN GO TO 670
660 PRINT AT y,x,CHR$ 16+CHR$ 3;CHR$ 144: BEEP 0.01,20: PRINT AT y,x,CHR$ 17+CHR$ 4,CHR$ 32: GO TO 680
670 PRINT AT y,x,CHR$ 16+CHR$ 3;CHR$ 144: BEEP 0.02,-15: PRINT AT y,x;CHR$ 20+CHR$ 1;CHR$ 16+CHR$ 2;CHR$ 32
680 NEXT x
690 NEXT y
700 REM

*****
*                               *
*  LOGO                       *
*                               *
*****

710 RESTORE 770
720 FOR x=7 TO 24
730 READ dato
```


terior del Spectrum estaba pintada en verde, se pulsa la tecla «EXTEND MODE» y a continuación, cuando el cursor de modo cambia a **E**, se pulsa la tecla seleccionada.

Para acceder a la sentencia inmediatamente inferior, pintada de rojo en el otro modelo, se pulsa «EXTEND MODE» y seguidamente, junto con la tecla seleccionada, cualquiera de las teclas «SYMBOL SHIFT». Cuando termina de visualizarse una sentencia en modo extendido, el cursor parpadeante cambia al modo anterior, **L O C**.

Modo **E**

Para pasar al modo **E** (Graphics Mode), el «Spectrum +» tiene la tecla «GRAPH». Pulsando esta tecla se tiene acceso a los semigráficos situados en las teclas con los números «1» a «0» y a los gráficos definidos por usuario, en las teclas con las letras de la «A» a la «V». Para retornar al modo anterior, es necesario pulsar de nuevo la tecla «GRAPH».

Edición de programas

Para la edición de programas y corrección de errores, el «Spectrum +» tiene teclas independientes con la misma funcionalidad que en el modelo anterior. Estas teclas son «EDIT», «DELETE» y los controles de cursor (arriba, abajo, izquierda y derecha), situados en este modelo a los lados de la barra espaciadora.

Con esta disposición se hace más agradable y rápida la edición y corrección de programas, ya que con sólo pulsar una tecla se consigue la función deseada.

16 MICROBASIC

```
740 PRINT AT 20 x;CHR$ 17+CHR$
4,CHR$ 16+CHR$ 0,CHR$ dato
750 BEEP 0.05,x
760 NEXT x
770 DATA 145,146,147,148,149,15
0
780 DATA 32,90,88,32,83,112,101
,99,116,114,117,109
800 BORDER 4
999 STOP
1000 REM
```

```
*****
*
*  GRAFICOS
*
*****
```

```
1005 RESTORE 1050
1010 FOR d=USR "a" TO USR "g"+7
1020 READ dato
1030 POKE d,dato
1040 NEXT d
1050 DATA 195,36,126,90,126,165,
169,129
1060 DATA 0,0,0,254,128,254,2,25
4
1070 DATA 0,128,0,191,160,160,16
0,160
1080 DATA 0,0,0,191,160,160,160,
191
1090 DATA 0,32,32,175,32,47,40,1
75
1100 DATA 0,8,0,235,42,234,42,23
4
1110 DATA 0,0,0,248,0,0,0,0
1120 RETURN
```

FUNCIONES DE CONTROL UTILIZADAS EN LA CORRECCION DE ERRORES

EDIT	Permite modificar la línea que contenga el prompt «>»
DELETE	Borra el carácter situado a la izda. de cursor.
↑	Despaza el prompt hacia la línea superior.
↓	Desplaza el cursor hacia la línea inferior.
→	Desplaza el cursor de edición hacia la derecha.
←	Desplaza el cursor de edición hacia la izda.

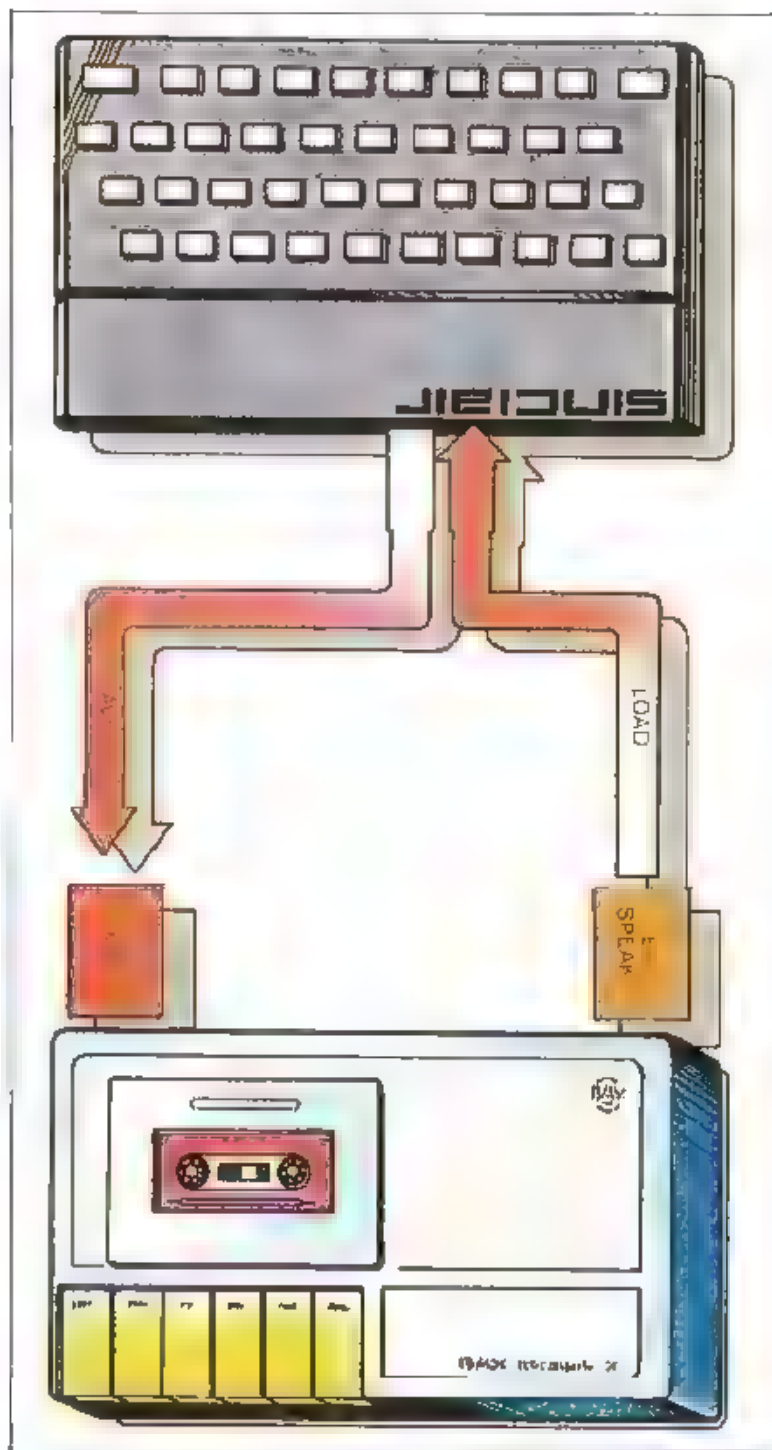
ALMACENAMIENTO DE PROGRAMAS

Mientras no desconecte el ordenador, podrá ejecutar cuantas veces quiera el programa almacenado en memoria por ejemplo, el propuesto en el capítulo anterior. Si desea volver a ejecutarlo en alguna otra ocasión, no parece lógico volver a tecarlo o tener el ordenador enchufado para que el contenido de la memoria no se pierda, por este motivo, los ordenadores personales tienen algún sistema de almacenamiento de programas. El Spectrum tiene la posibilidad de hacerlo en cintas de *cassette* comerciales, en cartuchos para *Microdrive* o en discos flexibles, también conocidos como *Diskettes*; en esta ocasión sólo vamos a tratar el almacenamiento en cintas de cassette.

Una vez que tengamos editado el programa, y sepamos con certeza que éste se ejecuta correctamente, pasaremos a grabarlo. Antes, es necesario conectar el cassette según se indica en el capítulo 5 de Manual de Instrucciones de Spectrum.

Para efectuar la grabación es necesario hacer uso del comando **SAVE**, que tiene como argumento el nombre que deseemos poner al programa. Debe ir entre comillas y no superar la cantidad de diez caracteres, éstos pueden ser letras, números o símbolos.

Pongamos un ejemplo, que queramos grabar el programa del ejercicio anterior y que de-



Grabación/recuperación.

seamos llamarle "EJER/1", la estructura de la instrucción sería.

SAVE "EJER/1"

para ejecutarla es necesario pulsar la tecla ENTER. Si hubiéramos asignado otro nombre al programa y nos apareciera el mensaje:

F Invalid file name, Ø:1

significaría que el nombre es más largo de diez caracteres o que lo intentamos almacenar como cadena vacía; es decir:

SAVE ""

Si no aparece el mensaje de error, es porque el nombre asignado es correcto, y en su lugar aparecerá el mensaje:

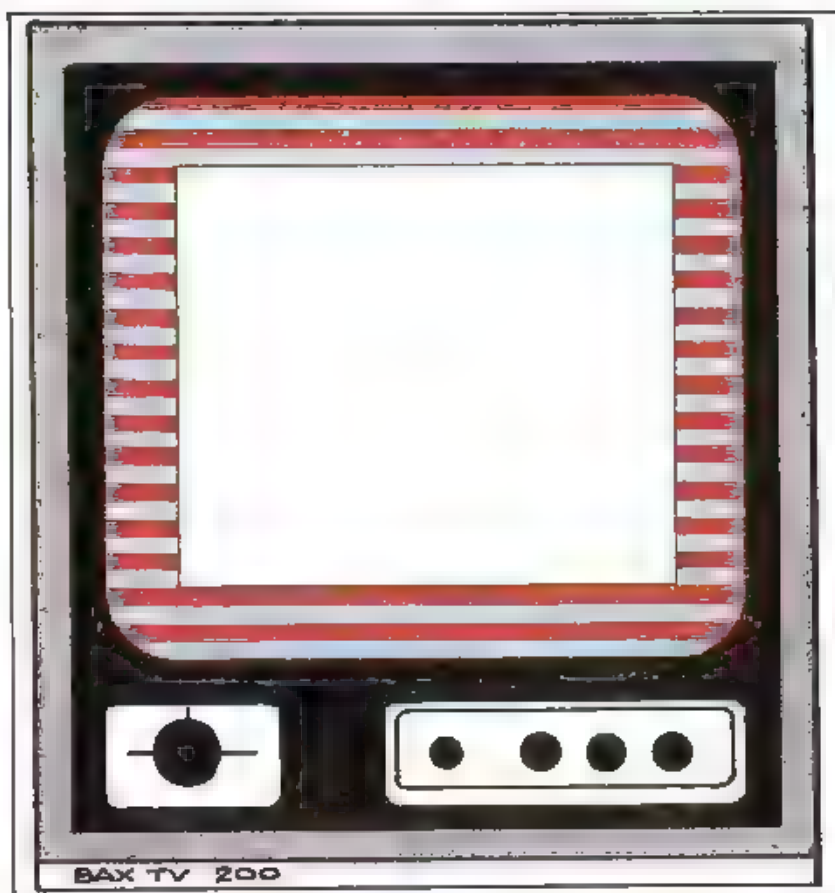
Start tape, then press any key

Este mensaje indica que se ponga el aparato de cassette en posición de grabar, es decir, pulsando las teclas PLAY y RECORD; si la tecla RECORD no entrara, es que la cinta utilizada está protegida contra posibles grabaciones, por tanto, es necesario utilizar otra cinta que no lo esté.

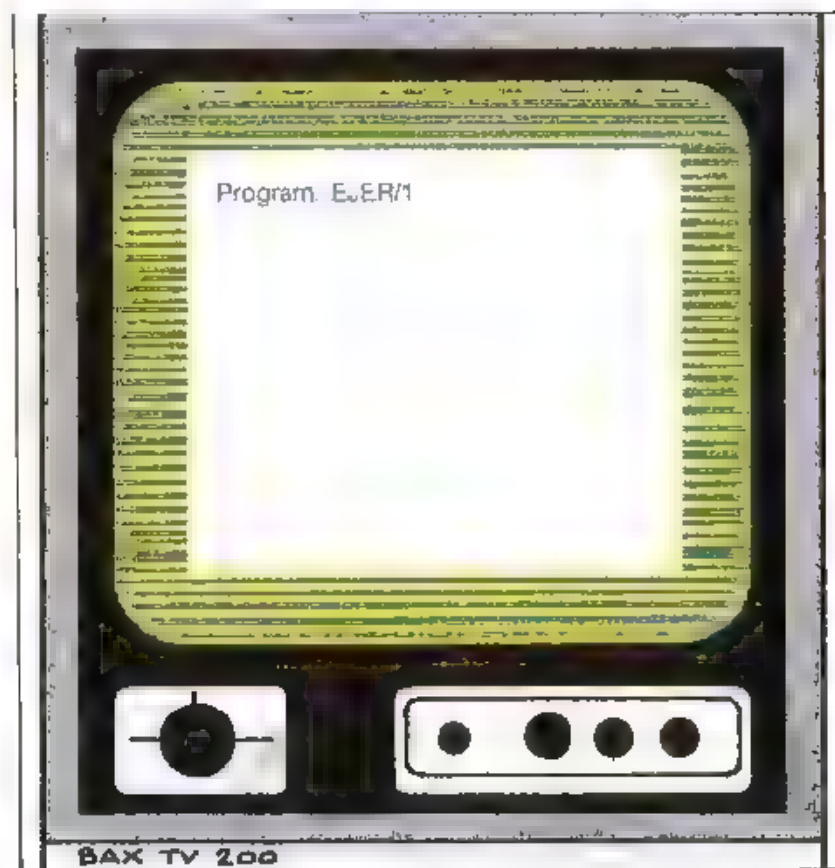
Cuando el aparato esté en marcha, pulsar cualquier tecla del Spectrum, excepto CAPS SHIFT o SYMBOL SHIFT, y la grabación empezará a efectuarse.

Mientras tanto, en el contor no de la pantalla se verán configuraciones de bandas coloreadas horizontales, que se van desplazando. Cuando la grabación termina, el ordenador envía el mensaje Ø OK, Ø : 1; en ese instante el aparato de cassette debe pararse. Hay

18 MICROBASIC



Señal de preajuste del nivel de grabación.



Señal de «grabación/recuperación».

que poner especial atención en no empezar a grabar en la zona transparente del comienzo de la cinta, ya que en esa zona no se puede grabar

Verificación

Antes de realizar cualquier otra tarea, es necesario cerciorarse de que el programa ha sido correctamente grabado en la cinta; para ello, se utiliza la sentencia VERIFY. Esta compara lo que hay grabado en la cinta con el contenido de la memoria.

Para verificar el programa, es necesario rebobinar la cinta de cassette hasta un punto anterior al comienzo de la grabación; para este fin es útil contar con un aparato de cassette que disponga de contador.

Utilizando el ejemplo ante-

rior, la estructura de la instrucción sería la siguiente:

VERIFY "EJER1"

Al pulsar ENTER, el contorno de la pantalla cambiará de color alternativamente. A partir de este momento, ya se puede poner en marcha el cassette pulsando la tecla PLAY. Cuando la cabecera del programa, es decir el principio, sea encontrado, aparecerá en la pantalla el siguiente mensaje:

Program: EJER1

y a continuación se verificará la grabación.

En el contorno de la pantalla se visualizarán las típicas configuraciones de bandas horizontales mencionadas anteriormente. Cuando la verificación ha sido efectuada el

ordenador nos enviará, si la grabación es correcta, el mensaje "Ø OK, Ø-1"

Si ésta no fuera correcta, pueden ocurrir dos cosas; primera, que la cabecera del programa no sea detectada, y por consiguiente, el ordenador se queda en un bucle sin fin esperando encontrarla. Lo segundo que puede ocurrir, es que la cabecera sea detectada, pero que algún dato del programa grabado no coincida con el contenido de la memoria. En este caso el ordenador nos enviará el mensaje

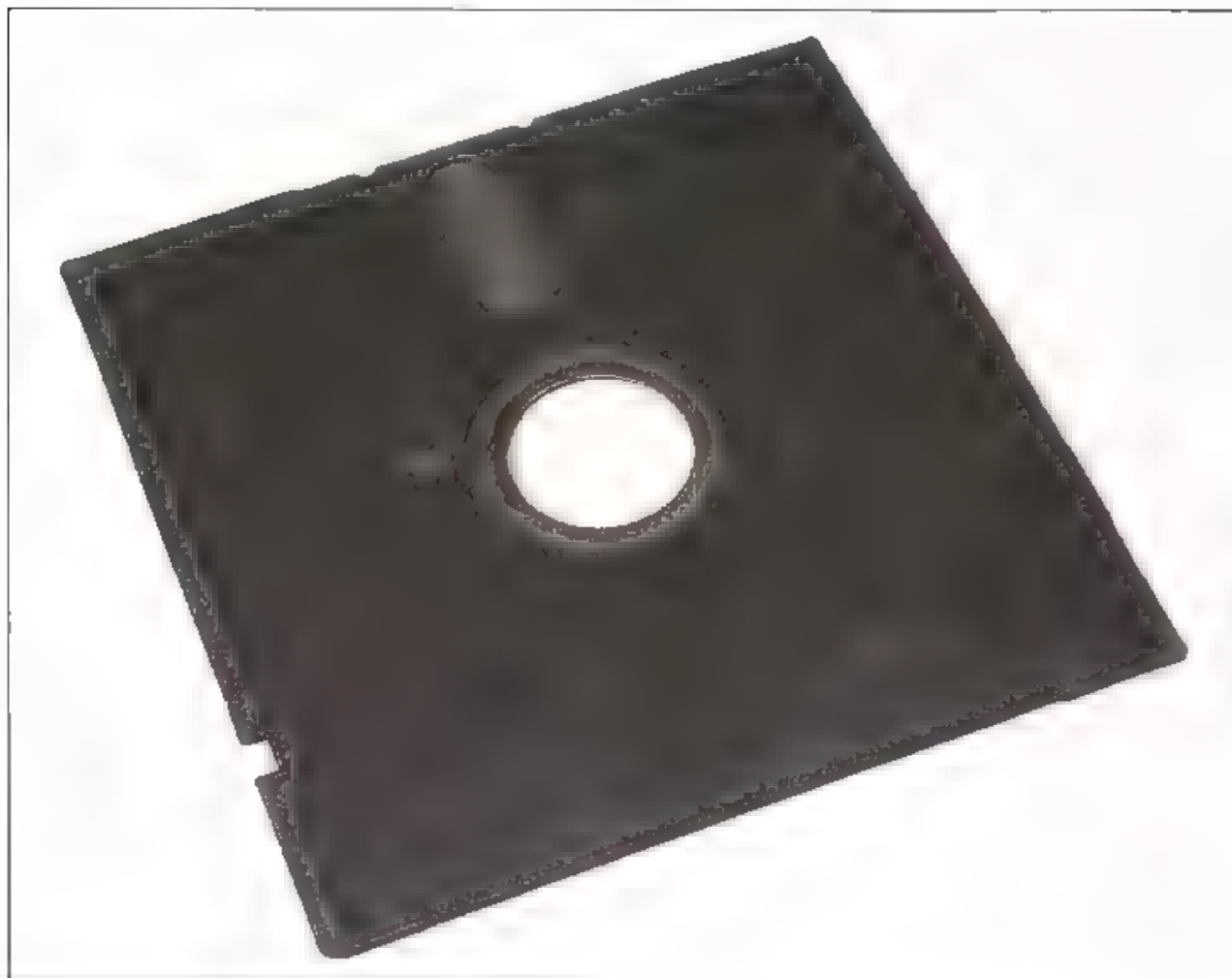
R Tape loading error. Ø 1

En ambos casos, volver a repetir la secuencia de verificación, y si el error persiste, grabar la cinta de nuevo.

Para salir del bucle sin fin del primer caso, es necesario hacer uso de la función



Almacenamiento en cinta o cassette.



Almacenamiento en Diskette.

BREAK, situada en la tecla espaciadora (SPACE); mantener esta tecla pulsada hasta que aparezca el mensaje:

D BREAK-CONT repeats. 0, 1

Una vez que esté grabado y verificado el programa, éste quedará almacenado *permanentemente* en la cinta, siempre y cuando no hagamos otra grabación en la misma zona y mientras la cinta sea conservada en las debidas condiciones.

Recuperación de programas

Si tenemos almacenado algún programa en cinta, podemos

volverlo a ejecutar cuando queramos; para ello es necesario copiar la información contenida en la cinta, en la memoria del ordenador, este proceso se conoce como *carga* o *recuperación* de programas.

El comando de carga es el denominado LOAD, éste, como es lógico, tiene por argumento el nombre del programa que deseamos recuperar. El asignar un nombre a un programa, con el comando SAVE, no fue una cosa caprichosa, sino que responde a una necesidad, cuando en una cinta tenemos grabados varios programas, uno a continuación de otro, la única forma de que el ordenador sepa cual debe cargar es

por el nombre que cada uno tiene asignado y que lo diferencia de los demás.

La estructura de la instrucción, siguiendo con el mismo ejemplo, es:

LOAD E.LER1"

Antes de poner el cassette en marcha, es necesario rebobinar la cinta hasta un punto anterior al comienzo del programa. Para empezar la carga, se necesita pulsar la tecla ENTER una vez introducido el comando y, a continuación, pulsar la tecla PLAY del aparato de cassette; a partir de este momento, tanto los mensajes de funcionamiento correcto, como los de error, son los mis-

mos que los proporcionados por la sentencia VERIFY.

Tanto con la sentencia VERIFY como con la LOAD, si la cinta se rebobinó hasta un punto lejano del comienzo del programa, al conectar el cassette irán apareciendo en pantalla los nombres de aquellos programas que el ordenador encuentre antes de llegar al especificado en el argumento de la sentencia.

Una vez que el programa está copiado en la memoria del ordenador, el aparato de cassette debe ser parado y la cinta puede retirarse para ser utilizada en otra ocasión.

Hay una variante de la sentencia LOAD, en la que el argumento es una cadena de caracteres vacía; con esta estructura el ordenador carga el primer programa que encuentre, aunque no se le especifique el nombre de éste; el formato es el siguiente:

LOAD ""

En este capítulo hemos explicado las estructuras básicas de las sentencias SAVE, VERIFY y LOAD; en otro posterior, se tratarán con más detalle.

Protección de programas

Cuando una cinta de cassette está grabada con programas definitivos, es conveniente proteger ésta contra posibles grabaciones accidentales.

Las cintas de cassette disponen en su parte posterior de dos lengüetas, una a cada lado, que sirven para indicar si la cinta es de lectura/escritura o por el contrario si es de sólo lectura. Cuando las lengüetas están intactas, la cinta permite ser grabada cuantas veces se quiera, por eso se dice que es de lectura/escritura,

en cambio, cuando son arrancadas, la cinta no permite posteriores grabaciones y por tanto es de sólo lectura.

Cuando la lengüeta A es arrancada, los programas de la cara 1 están protegidos contra escritura, si por el contrario es la B serán protegidos los de la cara 2. Para arrancar dichas lengüetas es conveniente la utilización de un destornillador a modo de palanca.

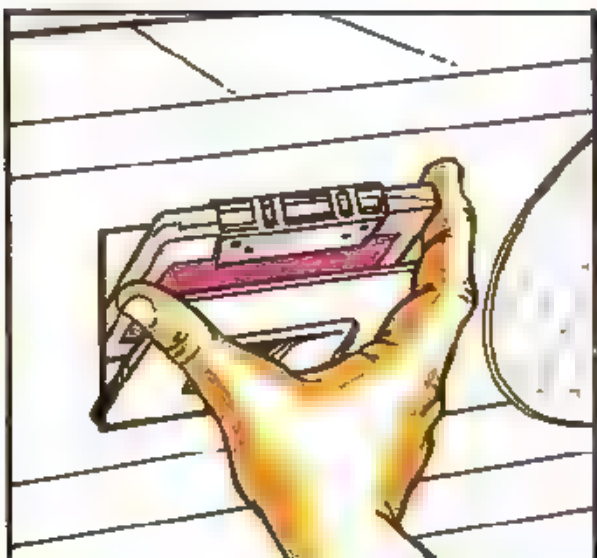
Si una nueva grabación fuese necesaria, los agujeros donde se encontraban las lengüetas deben ser tapados con cinta adhesiva.

ADVERTENCIA

Cuando se usan cintas de cassette protegidas contra escritura, la tecla RECORD de su cassette no permite ser pulsada. No intente forzarla, ya que podría dañar el aparato.



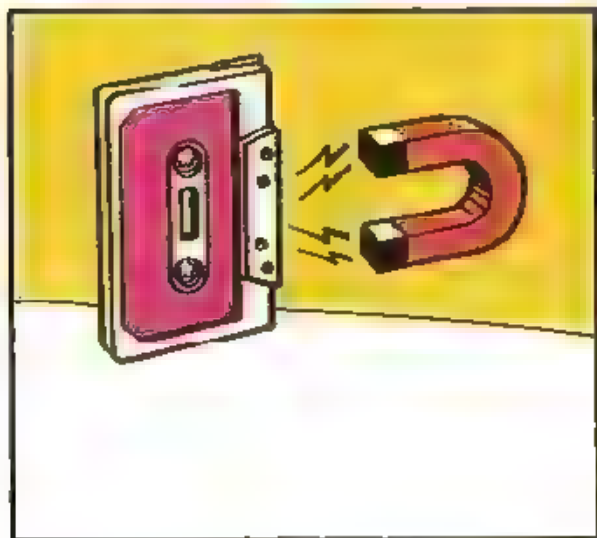
Almacenamiento en cartucho «Microdrive».



Inserte con cuidado.



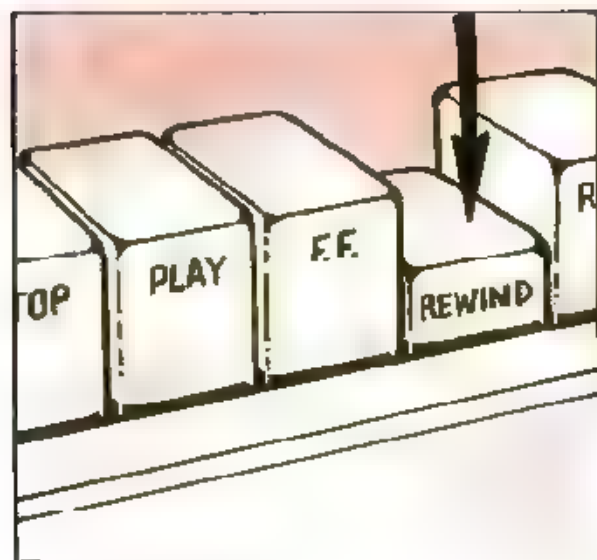
Conserve a 10° C. - 52° C./8%-80% Hr.



Evite campos magnéticos.



No tocar.



Rebobine al final.

22 MICROBASIC

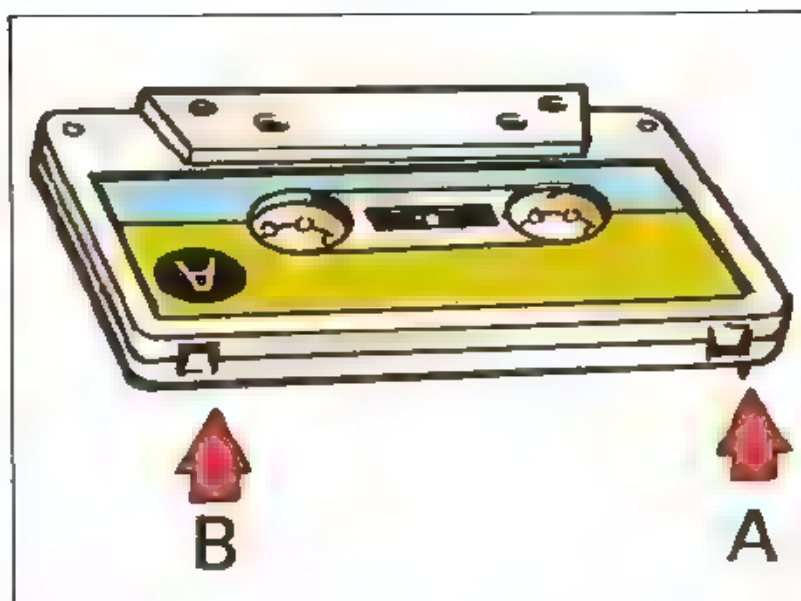


Conserve la cinta en su estuche.

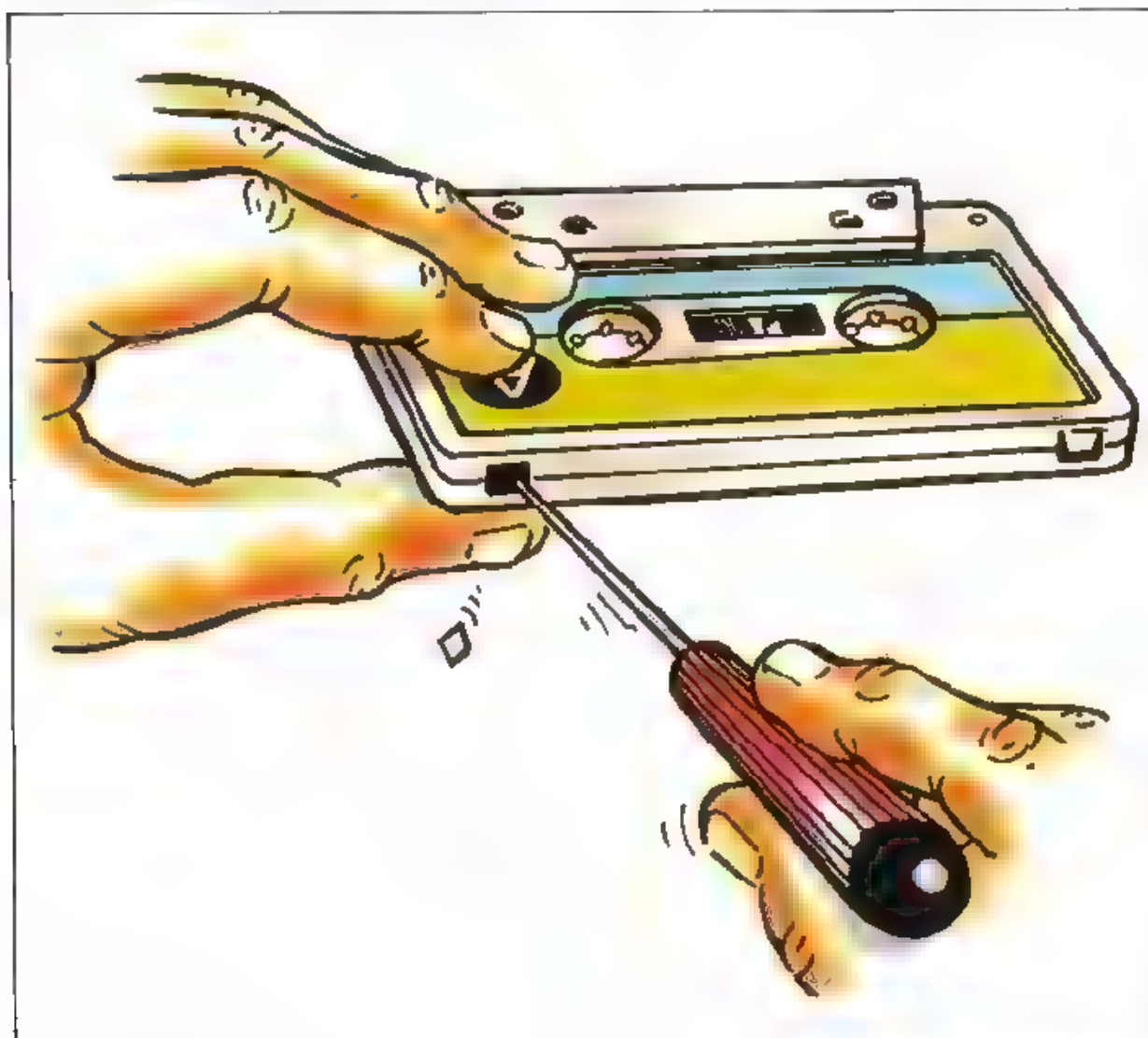
Conservación de cintas

Para no deteriorar los programas grabados, es necesario tener un cuidado especial con las cintas. Para una mejor conservación de éstas, conviene tener en cuenta los siguientes consejos:

El colocar una cinta en las proximidades de un televisor, puede traer graves consecuencias, ya que éstos, internamente, llevan incorporados transformadores y bobinas de alta tensión que generan *campos magnéticos*. Si son lo suficientemente potentes pueden alterar la información contenida



Lengüetas.



Protección de programas.

en la cinta, con la siguiente destrucción de los programas.

Oscilación

La mayoría de los aparatos de cassette comerciales disponen de un *circuito monitor*, que permite al usuario escuchar en el altavoz la grabación que se está efectuando desde

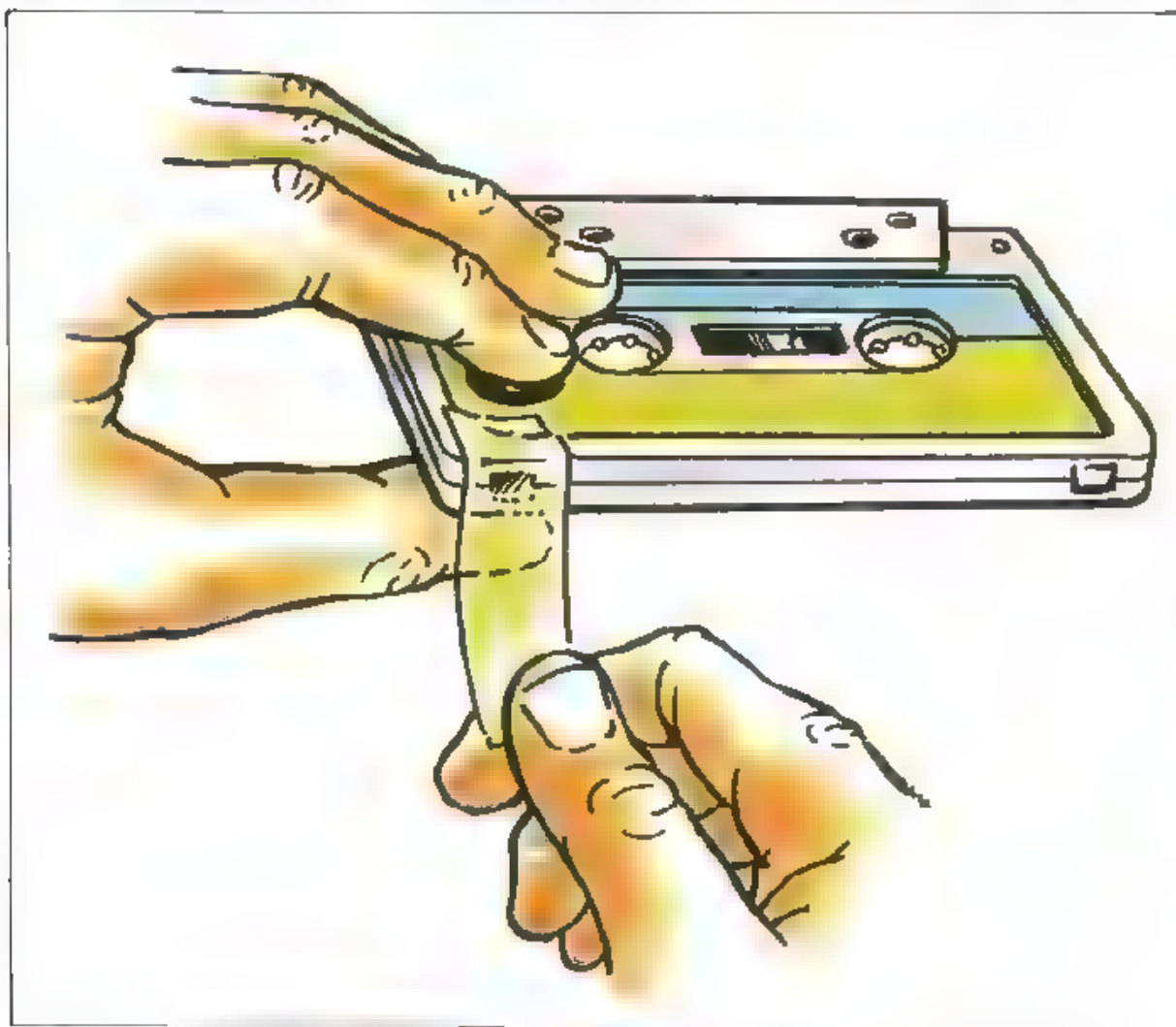
la de grabación se verá alterada, impidiendo que el programa pueda ser cargado posteriormente.

La solución consiste en abrir el bucle. Para ello desconecte cualquiera de las dos clavijas que unen la entrada EAR con la salida de auricular o altavoz externo.

té bien grabada, pueden presentarse dos situaciones:

—Que el nivel de grabación de la cinta sea distinto al de reproducción. Para subsanar esta anomalía, basta con aumentar ligeramente el volumen del cassette y proceder a cargar de nuevo.

—Que el ajuste de la cabe-



Tapado de lengüetas.

una radio, desde otro cassette o desde un plato giradiscos. Cuando desea grabar un programa y conecta los dos pares de clavijas (EAR y MC), la señal de grabación procedente del ordenador puede retornar de nuevo a él debido al circuito monitor. Este bucle formado empezará a oscilar, y la se-

Ajuste

Si emplea una cinta grabada en un aparato de cassette distinto al que maneja, puede ocurrir que no le sea posible cargar un programa, ya que aparece el correspondiente mensaje de error.

Suponiendo que la cinta es-

ta grabadora/reproductora fuera distinto. Para reajustar su cassette, utilice un destornillador del tipo estrella y retoque la altura de la cabeza hasta conseguir que el programa se cargue sin errores. Si desea volver al ajuste original, inserte una cinta grabada en ese cassette y retoque de nuevo

Constantes y variables

Los datos que el Spectrum procesa son de dos tipos: *constantes y variables*; ambos forman parte del argumento de algunas instrucciones.

Las constantes son datos que, durante la ejecución de un programa, no varían su valor, mientras que las variables pueden tomar distinto valor. La forma de representar una constante es por su valor; sin embargo, las variables se indican con un nombre simbólico.

Los datos también pueden dividirse en *numéricos y string* o *cadena de caracteres*. Podemos tener, por tanto, las siguientes combinaciones.

DATOS	
CONSTANTES	NUMERICAS
	STRING
VARIABLES	NUMERICAS
	STRING

Veamos unos ejemplos:

a) La fórmula que calcula el área de una superficie esférica es:

$$S = 4 \pi r^2$$

Como constantes numéricas tenemos el número "4", el exponente del radio (r) "2" y el valor de pi (π) "3,1415927"; las variables numéricas están indicadas por los símbolos " r " (radio) y " S " (superficie), que pueden tomar distintos valores durante la ejecución de un programa.

b) El argumento de la instrucción que imprime "El autor de RIMAS y LEYENDAS es BECQUER", es:

"El autor de " ;L\$;" es "; N\$

las constantes de cadena, son los string:

```

10 REM "CURSO BASIC/SINCLAIR"
20 REM
25 POKE 23658,8
30 BORDER 2: PAPER 2: INK 7: CLS
40 DRAW 0,175: DRAW 255,0: DRAW 0,-175: DRAW -255,0
42 PLOT 48,135: DRAW 0,-15: DRAW 15,0: DRAW 0,3: DRAW -12,0: DRAW 0,9: DRAW 12,0: DRAW 0,3: DRAW -15,0
44 PLOT 72,135: DRAW 0,-15: DRAW 15,0: DRAW 0,15: DRAW -15,0: PLOT 75,132: DRAW 0,-9: DRAW 9,0: DRAW 0,9: DRAW -9,0
46 PLOT 96,135: DRAW 0,-15: DRAW 15,0: DRAW 0,23: DRAW -3,0: DRAW 0,-8: DRAW -11,0: PLOT 99,132: DRAW 0,-9: DRAW 9,0: DRAW 0,9: DRAW -9,0
48 PLOT 120,135: DRAW 0,-15: DRAW 15,0: DRAW 0,3: DRAW -12,0: DRAW 0,3: DRAW 12,0: DRAW 0,9: DRAW -15,0: PLOT 123,132: DRAW 0,-3: DRAW 9,0: DRAW 0,3: DRAW -9,0
50 PLOT 144,143: DRAW 0,-23: DRAW 15,0: DRAW 0,15: DRAW -12,0: DRAW 0,8: DRAW -3,0: PLOT 147,132: DRAW 0,-9: DRAW 9,0: DRAW 0,9: DRAW -9,0
52 PLOT 168,135: DRAW 0,-15: DRAW 15,0: DRAW 0,3: DRAW -12,0: DRAW 0,12: DRAW -3,0: PLOT 168,143: DRAW 0,-4: DRAW 3,0: DRAW 0,4: DRAW -3,0
54 PLOT 192,135: DRAW 0,-15: DRAW 3,0: DRAW 0,12: DRAW 9,0: DRAW 0,-12: DRAW 3,0: DRAW 0,15: DRAW -15,0
56 PRINT FLASH 1;AT 12,9;"PARE LA CINTA"
58 PLOT 0,23: DRAW 255,0
60 PRINT AT 20,1;" ";
62 FOR X=1 TO 28
64 READ A
66 PRINT CHR$ A; . BEEP 0.05,A/2
68 NEXT X
70 DATA 127,32,77,73,67,82,79,72,79,66,66,89,32,38,32,82,65,70,65,69,76,32,80,82,65,68,69,83
72 PRINT AT 12,9;" "
74 PRINT #0;AT 1,1;"PULSE UNA TECLA PARA CONTINUAR"
76 PAUSE 0
78 BEEP 0.2,20
100 BORDER 1: PAPER 1: INK 7: CLS
200 REM
201 REM "PLANTILLA"
202 REM
210 DRAW 0,175: DRAW 255,0: DRAW 0,-175: DRAW -255,0
212 PLOT 0,151: DRAW 255,0
218 PRINT AT 1,1;" ";

```



```

220 RESTORE
222 FOR X=1 TO 20
224 READ A
226 PRINT CHR$(A);
228 NEXT X
230 PLOT 63,128
240 DRAW 129,0: DRAW 0,-9: DRAW
-129,0: DRAW 0,9
250 PRINT AT 6,12;"BINARIO"
260 PLOT 63,103
270 DRAW 128,0: DRAW 0,-24: DRA
W -128,0: DRAW 0,24
280 FOR X=71 TO 183 STEP 8
290 PLOT X,103: DRAW 0,-24
300 NEXT X
310 PLOT 63,48
340 DRAW 57,0: DRAW 0,-9: DRAW
-57,0: DRAW 0,9
360 PLOT 63,26
370 DRAW 57,0: DRAW 0,-13: DRAW
-57,0: DRAW 0,13
380 PRINT AT 16,8; OVER 1;"DECI
MAL"
470 REM
480 REM "ENTRADA DE DATOS"
490 REM
500 LET N$=""
510 PRINT #0;AT 1,0; FLASH 1;">
"; FLASH 0;" ";
520 FOR X=1 TO 6
530 PAUSE 0
540 LET A$=INKEY$
550 IF X<>1 THEN GO TO 700
560 IF A$="-" THEN LET S$=A$: P
RINT #0,S$; GO TO 745
570 IF A$>"0" AND A$<="9" THEN
LET S$="": LET N$=A$: PRINT #0;
S$+N$; GO TO 745
580 GO TO 530
700 IF CODE A$=13 THEN BEEP 0.0
5,20: GO TO 800
710 IF A$>"0" AND A$<="9" THEN
GO TO 730
720 GO TO 530
730 PRINT #0;A$;
740 LET N$=N$+A$
745 BEEP 0.05,20
750 NEXT X
770 REM
780 REM "VERIFICACION"
790 REM
800 LET N=VAL (N$)
810 IF N<0 OR N>65535 THEN PRIN
T #0;AT 1,0;"Numero no valido.";
FOR X=1 TO 200: NEXT X: PRINT #
0;AT 1,0;" "; GO
TO 500
812 REM
814 REM "DECIMAL"
816 REM
818 PRINT #0;AT 1,0;"Espere un
momento por favor.";
820 PRINT AT 19,0,S$
830 LET Y=19
840 IF N<10 THEN LET X=14

```

'El autor de "y" es "

y como variables, los símbolos L\$ y N\$; éstos, respectivamen-
te, almacenan el título y el
autor del libro; en un momen-
to determinado podrían tomar
los valores L\$ = "EL QUIJO-
TE" y N\$ = "CERVANTES"; en
este caso el mensaje a impri-
mir sería.

El autor de EL QUIJOTE es
CERVANTES

Constantes numéricas

En el BASIC/Sinclair, los nú-
meros se representan en cua-
tro formatos.

- NOTACION ENTERA.
- NOTACION FRACCIONARIA.
- NOTACION EXPONENCIAL.
- NOTACION BINARIA

Notación entera

En este formato se repre-
sentan los números *enteros*,
que por definición son aque-
llos que no contienen parte
decimal.

Los números enteros siguen
las siguientes reglas:

- Pueden ser positivos y
negativos.
- Precisión máxima de
ocho dígitos.
El signo "+" o espacio
en blanco indica núme-
ro positivo y el signo
"—" negativo.

Por tanto, un número entero es-
tá comprendido dentro del ran-
go + 9999999 y — 9999999.

Ejemplos correctos:

- + 15934
- 823
- 9357
- 1
- + 12345678

Ejemplos incorrectos

- 12.347 (Número con de-
cimales).

```

850 IF N>=10 AND N<=99 THEN LET
x=13
860 IF N>=100 AND N<=999 THEN L
ET x=12
870 IF N>=1000 AND N<=9999 THEN
LET x=11
880 IF N>=10000 THEN LET x=10
890 PRINT AT y,x;N
900 REM
901 REM "BINARIO"
902 REM
910 DIM I(16)
915 LET Z=N
920 FOR Y=15 TO 0 STEP -1
930 IF Z>=INT (2^Y) THEN LET Z=
Z-INT (2^Y): LET I(Y+1)=1
940 NEXT Y
950 PRINT AT 10,6;S$
960 PRINT AT 10,7; OVER 1," ";
970 FOR X=16 TO 1 STEP -1
980 PRINT OVER 1;I(X);
990 NEXT X
1000 PRINT #0;AT 1,0;"Desea otro
calculo (S/N)
1010 PAUSE 0
1020 LET A$=INKEY$
1030 IF A$="S" THEN BEEP 0.2,20:
GO TO 1055
1040 IF A$="N" THEN BEEP 0.2,20:
STOP
1050 GO TO 1010
1055 PRINT AT 10,7; OVER 1," ";
1060 FOR X=16 TO 1 STEP -1
1070 PRINT OVER 1;I(X);
1080 NEXT X
1090 PRINT AT 10,6;" "
1100 PRINT AT 10,8;" "
1150 PRINT #0;AT 1,0;" "
1200 GO TO 500

```

935749103 (Tiene nueve dí-
gitos).
32G14 (Contiene un ca-
rácter no numé-
rico).

Notación decimal

Sirve para representar aque-
llos números que contienen
decimales. En BASIC, la "coma
decimal" se sustituye por el
"punto decimal", aunque su re-
presentación se denomine de
"coma fija". Al igual que los
números enteros, la precisión
de los datos de salida es de
ocho cifras significativas; sin
embargo, los datos introduci-
dos por teclado (entrada) pue-
den tener mayor longitud, ya

que el ordenador efectúa un
redondeo.

El campo de los números
decimales está comprendido
entre + 0.0001 y + 99999999
/- 0.0001 y - 99999999.
Cuando el dígito situado a la
izquierda del punto decimal es
cero, su representación puede
ser omitida; es decir, la cifra .15
es la misma que 0.15.

Ejemplos de números deci-
males:

- 3.4569035
+ 5.47
.37851
+ 3537591.2

Notación exponencial

Otra forma de expresar nú-
meros es mediante la notación

exponencial, utilizando las po-
tencias de base 10. Por ejem-
plo.

NUMERO	NOTACION EXPONENCIAL
43012	43.012×10^3
370000	37×10^4
93.23	9.323×10^2
93.23	9.323×10^1
1000000	1×10^6

El formato que utiliza el
Spectrum para representar la
notación exponencial, es el in-
dicado en la fig. 1

La *mantisa* es un número
de ocho dígitos de precisión
máxima.

La *e* (minúscula) o *E*
(mayúscula) es un símbolo
que indica que la base de po-
tenciación es decimal (10).

El *exponente* es un número
de dos dígitos de precisión
máxima, que expresa la poten-
cia a la que hay que elevar la
base.

Tanto la mantisa como el
exponente tienen sus propios
signos "+" o "-".

En el ejemplo de la figura 1,
podemos comprobar la nota-
ción + 3.7281234 E-14 es equi-
valente a $3.7281234 \times 10^{-14}$.
Desplazando la coma (punto
en BASIC) y rectificando por
tanto el valor del exponente,
podríamos tener muchas com-
binaciones y todas con el mis-
mo valor, por ejemplo:

1. 0.0003728 $\times 10^{-10}$
2. 0.0037281 $\times 10^{-11}$
3. 0.0372812 $\times 10^{-12}$
4. 0.3728123 $\times 10^{-13}$
5. 3.7281234 $\times 10^{-14}$

De todas éstas, el Spectrum
utiliza únicamente la 5, ya que
proporciona el mayor número
de cifras significativas en la

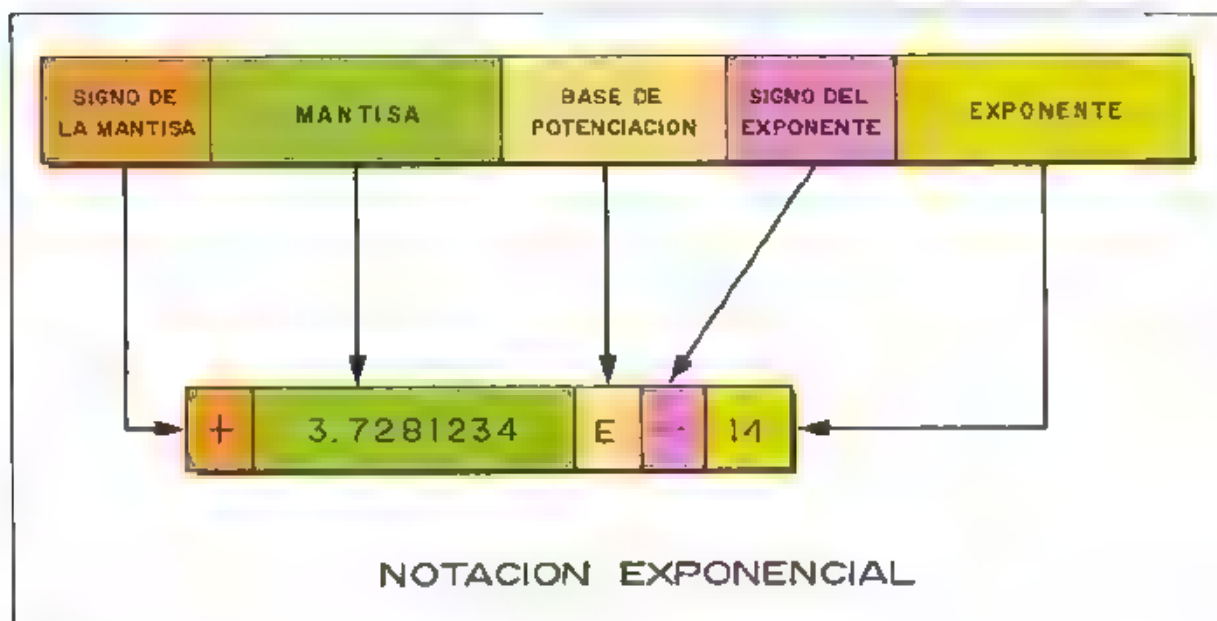


Fig. 1.

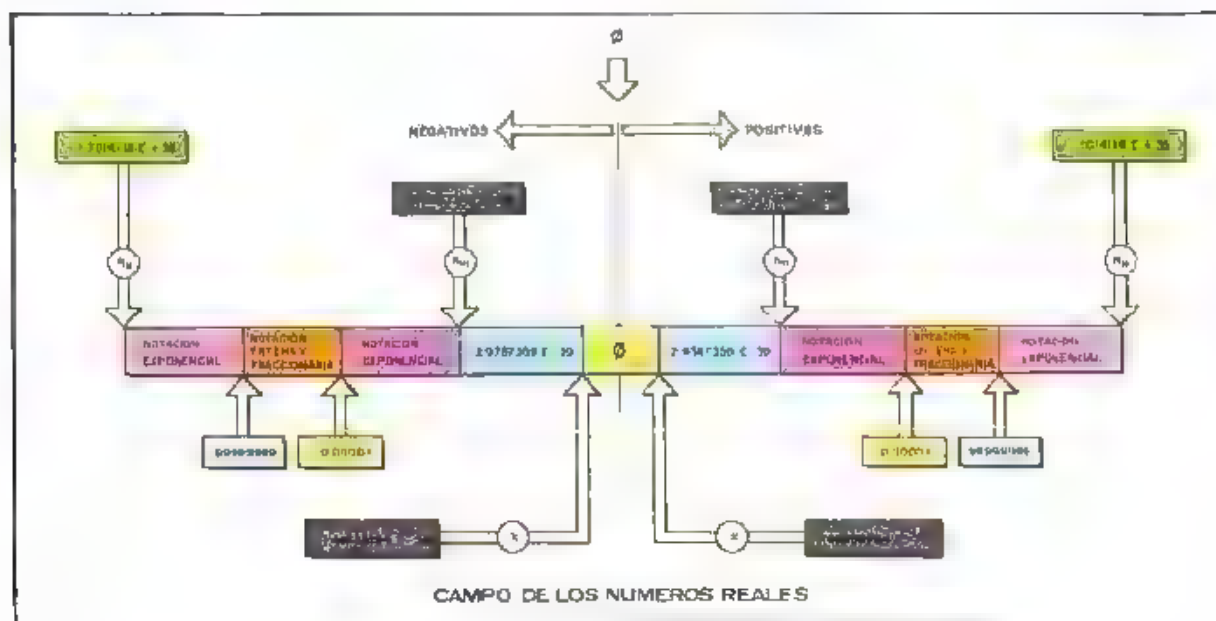


Fig. 2.

mantisa (un dígito distinto de cero, punto decimal y hasta siete dígitos más); este tipo de representación se denomina *coma flotante*.

La notación exponencial la utiliza el ordenador para representar números cuyo valor sea inferior a 0.00001 o superiores a 99999999, dentro del campo de los números positivos.

En la figura 2, se puede observar el campo de los núme-

ros reales; a la derecha del 0 se encuentran los números positivos y a la izquierda los negativos, ya que ambas partes son iguales, bastará con explicar una de ellas.

El mayor número positivo que se puede representar es "NM", cuyo valor es igual a 1.7014118 E + 38; cuando el ordenador realiza una operación y el resultado es superior

a "NM", se presenta en pantalla el mensaje:

6 Number too big, a : b

a = línea de programa donde se originó el error

b = n.º de instrucción dentro de la línea.

El menor número positivo es "Nm" y su valor es 2.9387359E - 39; cualquier

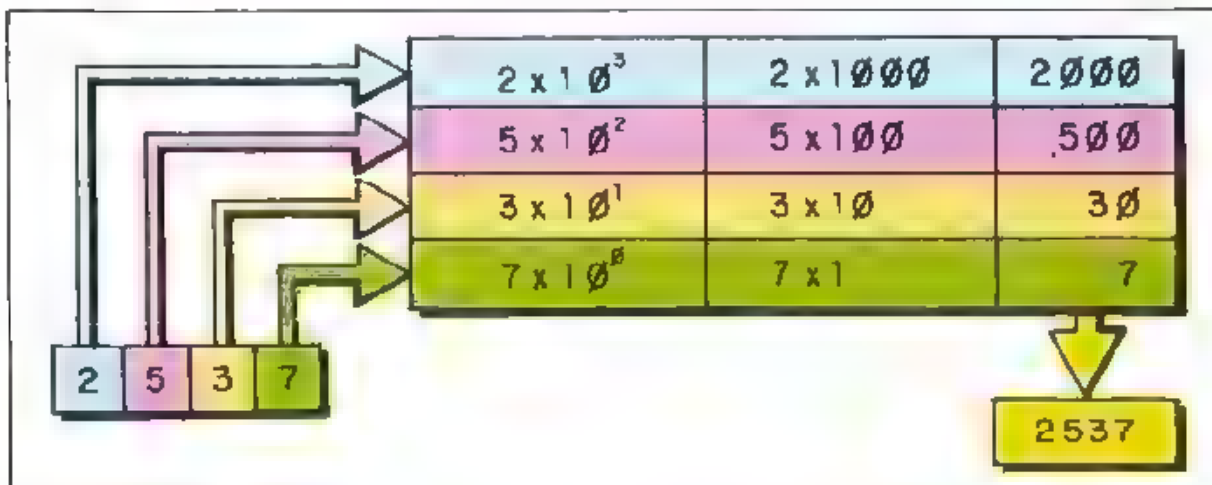


Fig. 3.

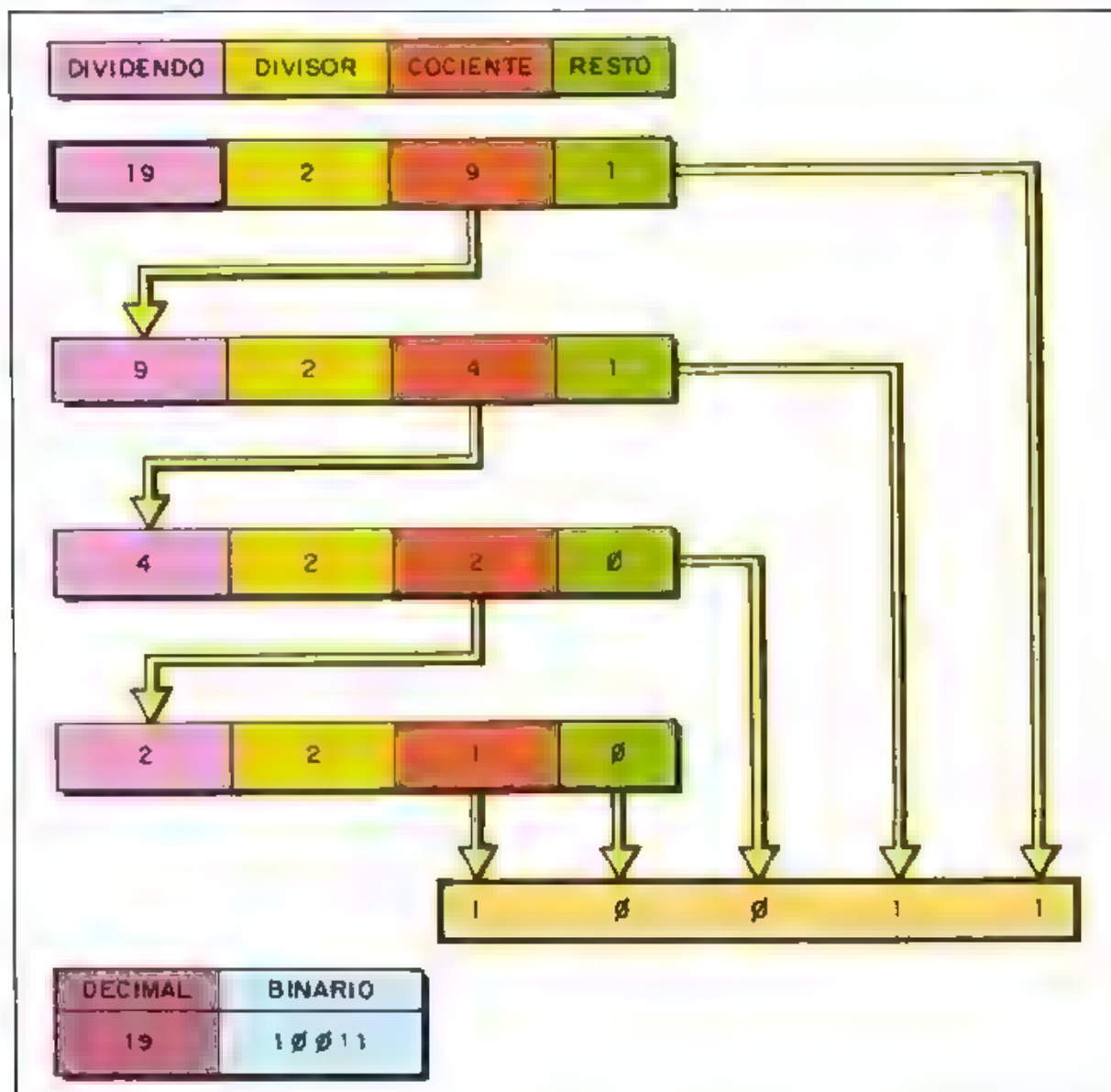


Fig. 4.

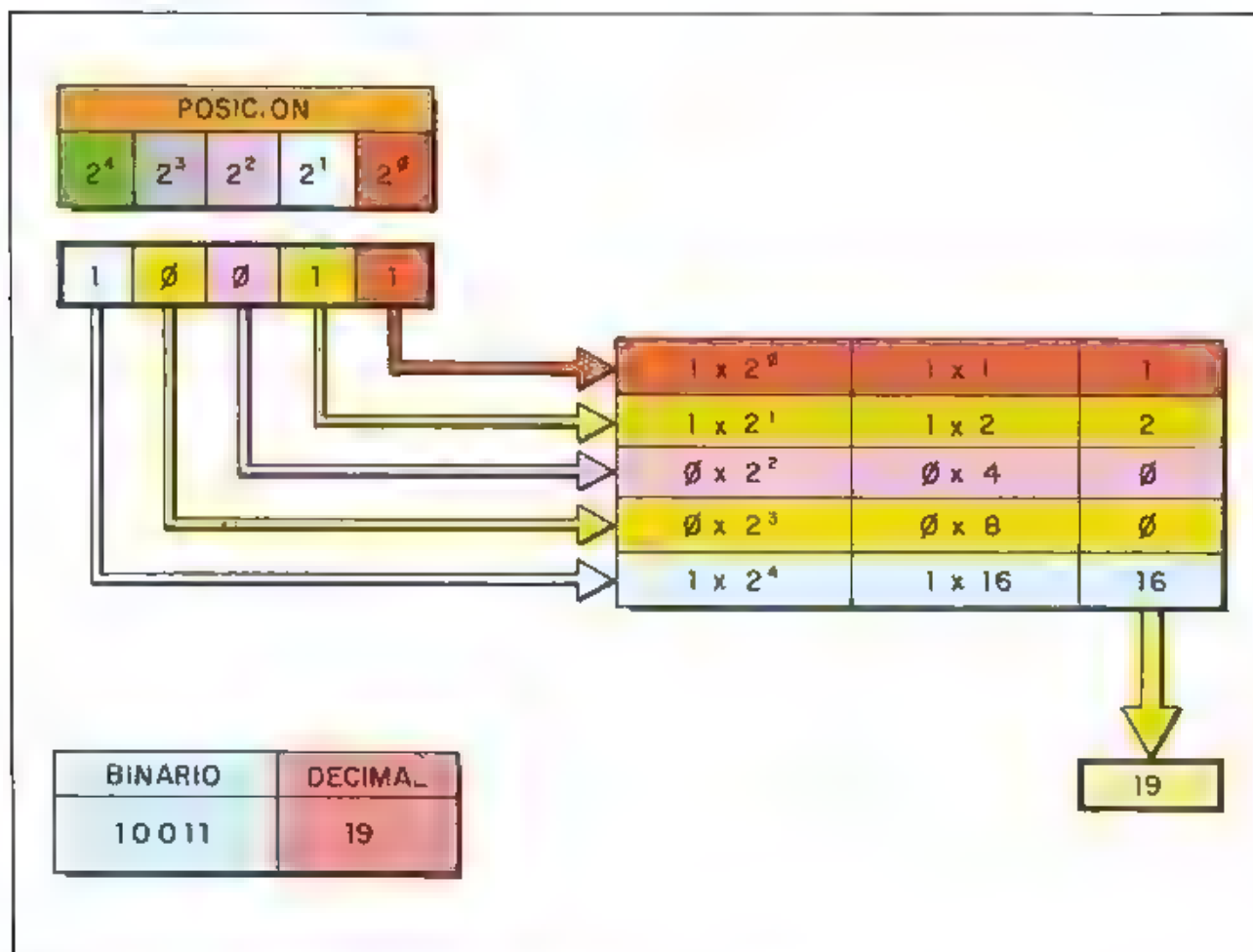


Fig. 5.

número inferior a éste y superior a "X" (1.46936794E - 39) será representado con el valor de "Nm". Valores inferiores a "X" serán tomados como 0.

Teclée el siguiente miniprograma y podrá comprobar cómo representa el Spectrum las constantes numéricas que le sean introducidas por el teclado.

```

10 INPUT "CONSTANTE NUMERICA?"
   "a
20 PRINT a
30 GO TO 10

```

Notación binaria

Para expresar un número, normalmente empleamos el sistema *decimal*, en el que se manejan diez símbolos (0, 1, 2, 3, 4, 5, 6, 7, 8 y 9) con los que se puede expresar cualquier

valor, utilizando las potencias de base 10 (fig. 3).

Cada número tiene un valor dependiendo de la posición que ocupa; por ejemplo, en el n.º 2537, el 2 ocupa la posición de las unidades de millar y tiene el valor 2000, el 5 las centenas (500), el 3 las decenas (30) y el 7 las unidades (7); sumando todos, obtenemos el valor 2537.

Otro sistema de numeración es el que utiliza internamente el ordenador y que se llama *binario*, ya que sólo utiliza dos símbolos, el 0 y el 1. Para expresar un valor en este sistema se utilizan las potencias de base 2; el valor de los símbolos también es opcional, ya que depende de la posición que ocupen.

En el Spectrum, los datos que deban ser introducidos en

notación binaria, tienen que ir precedidos de la palabra clave BIN, que se encuentra situada en la letra B. Los números binarios pueden ser positivos o negativos, pero en cualquier caso tienen que ser enteros, no se permiten números con parte decimal. El mayor valor expresado en BIN 111111111111111, que equivale al valor 65535 en decimal, y el menor, lógicamente, — BIN 111111111111111 (— 65535).

Ejemplos de notación binaria:

```

BIN 11010
BIN 001110
BIN 10
BIN 1111

```

Decimal-binario

Para transformar un número decimal en binario, observe



Portada del programa CODEBIN.

el gráfico de la figura 4 que lo explica con detalle. El n° a transformar es el 19, éste se divide entre dos, el cociente resultante se vuelve a dividir otra vez en dos, y así sucesivamente hasta que el último cociente sea 1; el número binario se forma con este cociente y con los restos obtenidos en las divisiones, tal como indica la figura.

Binario-decimal

Un número se transforma en su correspondiente decimal multiplicando cada símbolo (0 ó 1) por el valor de la posición que ocupa en base 2, sumando todos los valores se obtiene el decimal. La figura 5 representa gráficamente esta transformación.

Ejercicio

Si desea convertir un número binario en decimal, sin te-

ner que hacer cálculos engorrosos, utilice el siguiente comando directo para números positivos:

PRINT BIN "notación binaria"
y para los negativos:

PRINT -BIN "notación binaria"
ejemplos.

```
PRINT BIN 1010
PRINT -BIN 111101
```

Si, por el contrario, lo que desea es pasar un número decimal a binario, edite el programa que le proponemos en este capítulo.

Cuando lo salve en cinta, hágalo de la forma:

SAVE "Codebin" LINE 10

De esta manera se autoejecutará al utilizar el comando LOAD y no hará falta teclear RUN una vez cargado.

Realizada la presentación del programa en pantalla, pul-

sando cualquier tecla, se pasa a la ejecución del programa principal. En la parte inferior aparece un prompt parpadeante " " indicando que el ordenador está preparado para la introducción del número decimal. Para teclear un negativo, es necesario que vaya precedido por el signo " - ". Si el valor del dato fuera superior a ± 65535 , el ordenador nos presentará un mensaje de error y se visualizará de nuevo el prompt. Cuando el dato es correcto, aparece un mensaje de espera, ya que el ordenador necesita un tiempo para realizar los cálculos; finalizado éste, es presentado de forma binaria e n° deseado.

Para realizar un nuevo cálculo, puse la tecla "S" e introduzca el nuevo valor.

Constantes alfanuméricas

También son conocidas como *cadenas* o *strings*. Están

formadas por una secuencia de caracteres encerrados entre comillas; la estructura general es:

STRING
"cadena de caracteres"

Los caracteres pueden ser letras mayúsculas o minúsculas (a, z, L,...), números (0, 4, 9,...), caracteres especiales (@, #, \$,...) o comandos BASIC (VERIFY, BIN, OUT...).

Ejemplos:

- "Curso BASIC/
- SINCLAIR"
- "© MICROHOBBY
- & RAFAEL PRADES"
- "Enero tiene 31 días"
- "3,1415927 es el valor de PI"

Cuando una cadena no tiene ningún carácter ("") se denomina *vacía* o *nula*, la cadena (" ") no se considera vacía, ya que contiene el carácter correspondiente al espacio.

Si dentro de una cadena ha de ir incluido el carácter de las comillas ("), éste deberá te-erse por duplicado, como indicador de que no es el final de la cadena.

Ejemplos:

- "El ""Spectrum"" es un ordenador personal"
- "El significado de ""Anticuerpo"" es..."

Si visualizamos estas dos cadenas, anteponiendo el comando PRINT, observaremos que las comillas no aparecen repetidas.

Variables numéricas

Este tipo de variables está formada únicamente por letras y números, pudiendo estar constituida su longitud por distinta cantidad de caracteres. El primer carácter debe ser obligatoriamente una letra.

Ejemplos.

- Color
- V12
- puntuación
- x1p12

No existe ninguna diferencia entre las variables escritas en letras mayúsculas o minúsculas; los siguientes ejemplos se refieren a la misma variable escrita de distintas formas.

- FJERZA
- Fuerza
- fuerza
- FuErZa

En ocasiones, para facilitar una posterior interpretación del significado de la variable, puede introducirse el carácter *espacio* tantas veces como se quiera. No existe ninguna diferencia entre una variable escrita con espacios y otra que no lo esté.

Ejemplos:

- RESISTENCIA
- DEL AIRE
- Resistenciadel Aire
- Resistenciadelaire

El valor asignado a una variable de tipo numérico debe ser una constante u otra variable, ambas, lógicamente, de tipo numérico.

Ejemplos:

- KILO = 12
- PESO = KILO
- Potencia = 30
- grados = 27
- Temperatura = grados

Variables alfanuméricas

Las variables alfanuméricas o de cadena, están constituidas por una sola letra, mayúsculas o minúsculas, indistintamente, seguida del símbolo *dólar* "\$".

Ejemplos:

- M\$
- J\$
- X\$
- T\$

El valor asignado a una variable alfanumérica debe ser, o una constante o una variable, ambas del tipo *cadena alfanumérica*.

Ejemplos:

- S\$ = "Producto"
- T\$ = S\$
- K\$ = "1024 Kbytes"
- N\$ = "3518E + 14"

OPERADORES

Los operadores son símbolos que expresan el tipo de operación que ha de realizarse, bien entre dos constantes, bien entre una variable y una constante, etc. Veamos unos ejemplos.

OPERANDO	OPERADOR	OPERANDO
3	+	1200
valor	=	100
A\$	<>	"Antonio"
527	AND	32

Existen tres tipos de operadores.

- ARITMETICOS.
- DE RELACION.
- LOGICOS.

Operadores aritméticos

Son aquellos que permiten ejecutar las operaciones aritméticas básicas: suma, resta, etc. Los símbolos utilizados son los indicados en la tabla.

OPERACION	SÍMBOLO
Suma	+
Resta	-
Multiplicación	*
División	/
Potenciación	↑

Ejemplos:

$10002 * 7$ (10002 por 7)
 $3 \uparrow 2$ (3 al cuadrado)
 $456 / 2$ (456 entre 2)

Expresiones aritméticas

Son conjuntos de constantes y variables unidas entre sí por operadores aritméticos.

Ejemplos:

$(3 + 5) / 7$
 $8 \uparrow 2 / 100 * 5$
 $gr * X \uparrow 3 / tz$
 $voltage \uparrow 2 / resistencia * tiempo$

Las variables utilizadas tienen que estar definidas previamente.

mente, ya que de lo contrario, cuando el ordenador realice un cálculo y alguna no lo esté, enviará el siguiente mensaje de error:

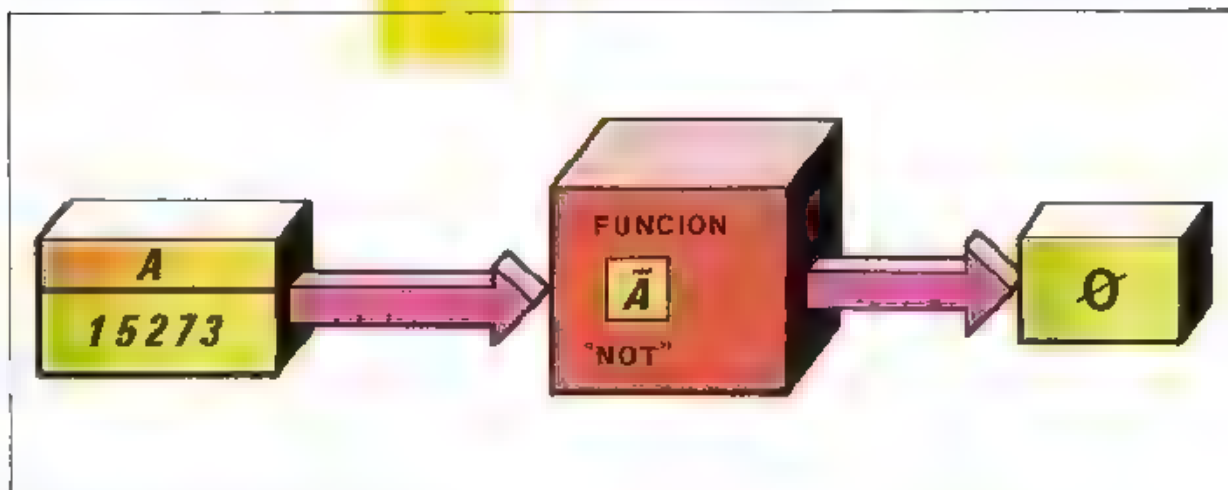
2 Variable not found

Al igual que en álgebra, se pueden utilizar los paréntesis, pero no las multiplicaciones implícitas, es decir, la expresión $(5 + X) (8 - Y)$, en BASIC, se escribe:

$(5 + X) * (8 - Y)$

Cálculo de expresiones

El ordenador cuando realiza un cálculo, lo hace siempre atendiendo al valor de *prioridad* que tenga cada operación. De las cinco operaciones aritméticas básicas, la potenciación tiene mayor prioridad que el resto, después la multiplicación y la división, ambas con el mismo valor, y por último, la suma y la resta, también con idéntica prioridad.



Función «AND». Estructura 1.

Cuando en una expresión hay dos operaciones de la misma prioridad, el ordenador efectúa los cálculos de izquierda a derecha.

Veamos por pasos cómo el ordenador calcula la siguiente expresión para $x = 3$ e $y = 2$:

- $16 + x + 3\phi - y / 2$
a) $16 + 3 + 3\phi - 2\phi / 2$
b) $16 + 9\phi - 2\phi / 2$
c) $16 + 9\phi - 1\phi$
d) $106 - 1\phi$

Resultado: 96

Utilizando los paréntesis se puede alterar el orden de evaluación de las operaciones, ya que éstos se calculan primero. Aprovechando la expresión anterior, vamos a observar que el resultado varía colocando los paréntesis en las operaciones de menos prioridad.

- $(16 + x) + (3\phi - y) / 2$
a) $(16 + 3) + (3\phi - 2\phi) / 2$
b) $16 + (3\phi - 2\phi) / 2$
c) $16 + 1\phi / 2$
d) $18\phi / 2$

Resultado: 9 ϕ

Operadores de relación

Permiten realizar las comparaciones entre operandos (constantes o variables) tanto numéricos como de cadena.

OPERADOR	SÍMBOLO
Igual	=
Distinto	<>
Mayor	>
Menor	<
Mayor o igual	>=
Menor o igual	<=

Las operaciones realizadas con estos operadores sólo tie-

nen dos resultados o soluciones, es decir:

— Si la condición impuesta por el operador se cumple, es decir, que es *verdadera* (true), el valor del resultado es «1».

— Si por el contrario, la condición no se cumple, es decir, que es *falsa* (false), el valor se vuelve «0».

Ejecute los siguientes comandos directos y compruebe lo explicado anteriormente.

COMANDO	CONDICIÓN	RESULTADO
PRINT 3 = 7	Falsa	0
PRINT 10 > 99	Verdadera	1
PRINT 8 <> 8	Falsa	0
PRINT 10 >= 7	Verdadera	1

El símbolo «—» también sirve para asignar un valor a una variable.

Operadores lógicos

Se utilizan para realizar las operaciones lógicas a nivel de expresión.

OPERACION	FUNCIÓN
Producto lógico	AND
Suma lógica	OR
Negación	NOT

En otras ocasiones, estos operadores se utilizan para relacionar dos expresiones mediante una condición, por ejemplo:

— Que el ordenador realice una determinada tarea si se cumplen varias relaciones.

RELACION	OPERADOR	RELACION
$x >= 0$	AND	$x <= 9$

En este caso sólo se cumplirá la condición cuando la variable x sea mayor o igual a 0 y a la vez sea menor o igual a 9, es decir, que esté comprendida entre 0 y 9.

Que se realice la tarea cuando simplemente alguna de las condiciones se cumpla.

RELACION	OPERADOR	RELACION
$a = 10\phi$	OR	$1 > 1$

La condición se cumple, bien cuando el valor de la variable a sea 10 ϕ , bien cuando el valor de 1 sea mayor que 1 ó bien cuando se cumplan ambas condiciones.

Función «AND»

La estructura de esta función es la que se muestra a continuación, siendo a y b dos expresiones numéricas.

$a \text{ AND } b$

Estas expresiones solamente pueden tomar los valores 0 o distinto de 0, este último lo vamos a representar como «<>0». Teniendo en cuenta todas las posibles combinaciones que pueden tomar a y b , vamos a mostrar su *tabla de verdad*:

a	b	RESULTADO
0	0	0
0	<>0	a(0)
<>0	0	0
<>0	<>0	a

de ésta se deduce que si el valor de b es igual a 0, el resultado de la función es 0, independientemente del valor de a . Si el valor de b es distinto de 0, la función asume el valor de a .

Ejemplos:

OPERACION	RESULTADO
132 AND \emptyset	\emptyset
8 \emptyset AND 1 \emptyset	8 \emptyset
\emptyset AND 25	\emptyset
1 AND 9 \emptyset	1

La función AND, también puede tomar la estructura:

a\$ AND b

en este caso el resultado de la función será una cadena vacía

las expresiones a y b sólo pueden tomar valores numéricos. Con todas las combinaciones se obtiene la siguiente tabla de verdad:

a	b	Resultado
\emptyset	\emptyset	a (\emptyset)
\emptyset	$\langle \rangle \emptyset$	1
$\langle \rangle \emptyset$	\emptyset a	a
$\langle \rangle \emptyset$	$\langle \rangle \emptyset$	1

Mirando detenidamente la tabla se observa que si b to

Función «NOT»

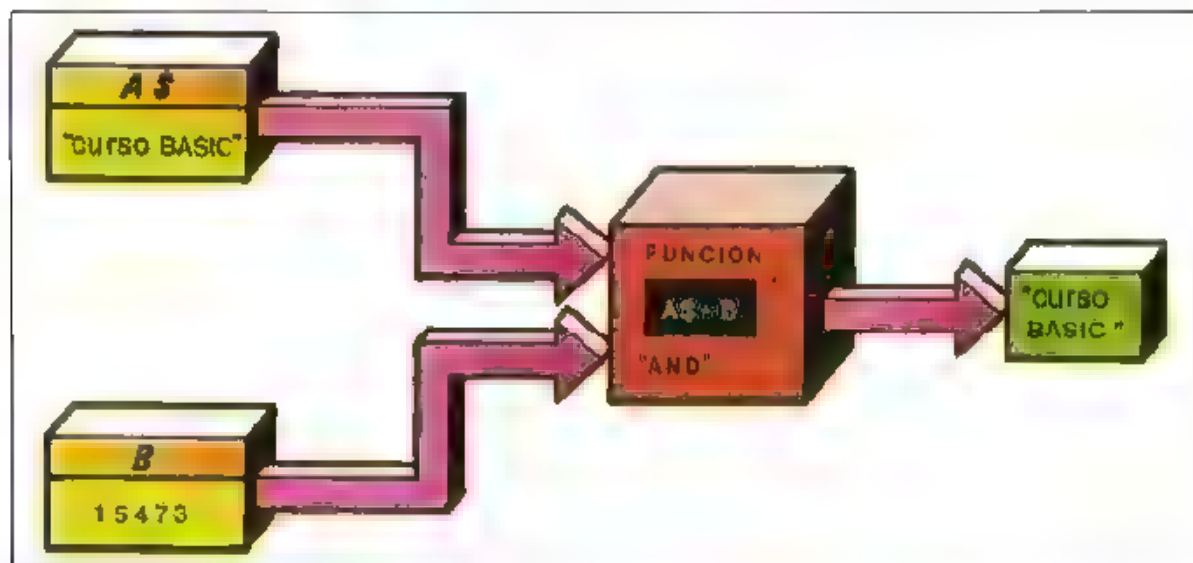
La función NOT afecta solamente a una expresión y su estructura es:

NOT a

su tabla de verdad es:

a	RESULTADO
\emptyset	1
$\langle \rangle \emptyset$	\emptyset

Como se puede apreciar, el resultado de la función es la



Función «AND». Estructura 2.

cuando b sea \emptyset y será la cadena a\$ cuando b sea distinto de \emptyset .

Ejemplos:

OPERACION	RESULTADO
«PEDRO» AND 3 \emptyset	«PEDRO»
«JJAN» AND \emptyset	«»

Función «OR»

La estructura de la función OR es la siguiente:

a OR b

ma el valor \emptyset , la función resultante adquiere el de a, sin embargo, si el valor de b es diferente de \emptyset , el valor de la función es 1, independientemente del que tenga a.

Ejemplo:

OPERACION	RESULTADO
\emptyset OR 3 \emptyset	1
2 \emptyset OR \emptyset	2 \emptyset
55 OR 7	1
\emptyset OR \emptyset	\emptyset

negación de la expresión a, es decir, vale 1 si a es igual a \emptyset y \emptyset si a es distinto de \emptyset . Ejemplo:

OPERACION	RESULTADO
NOT 7	\emptyset
NOT \emptyset	1

Anteponiendo el signo «—» a la función NOT, el resultado cambia de signo.

OPERACION	RESULTADO
NOT 4	\emptyset
— NOT \emptyset	—1

CARACTERES ASCII																			
C	u	r	s	o		B	A	S	I	C	/	S	I	N	C	L	A	I	R
↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
67	117	114	115	111	32	66	65	83	73	67	47	83	73	78	67	76	65	73	82
CODIGO DECIMAL																			

Función «OR».

Ejercicio

Estas funciones pueden en-
cadenarse para formar otra
más complicada. Haciendo
uso de los paréntesis «()» se
consigue facilitar su interpre-
tación; por ejemplo:

$$(X \text{ AND } Y) \text{ OR } (\text{NOT } X)$$

si asignamos a a variable X el
valor 3 y a Y el valor 5, veamos
cua. es el resultado final resol-
viendo la función por pasos:

a) $(3 \text{ AND } 5) \text{ OR } (\text{NOT } 3)$

b) $3 \text{ OR } (\text{NOT } 3)$

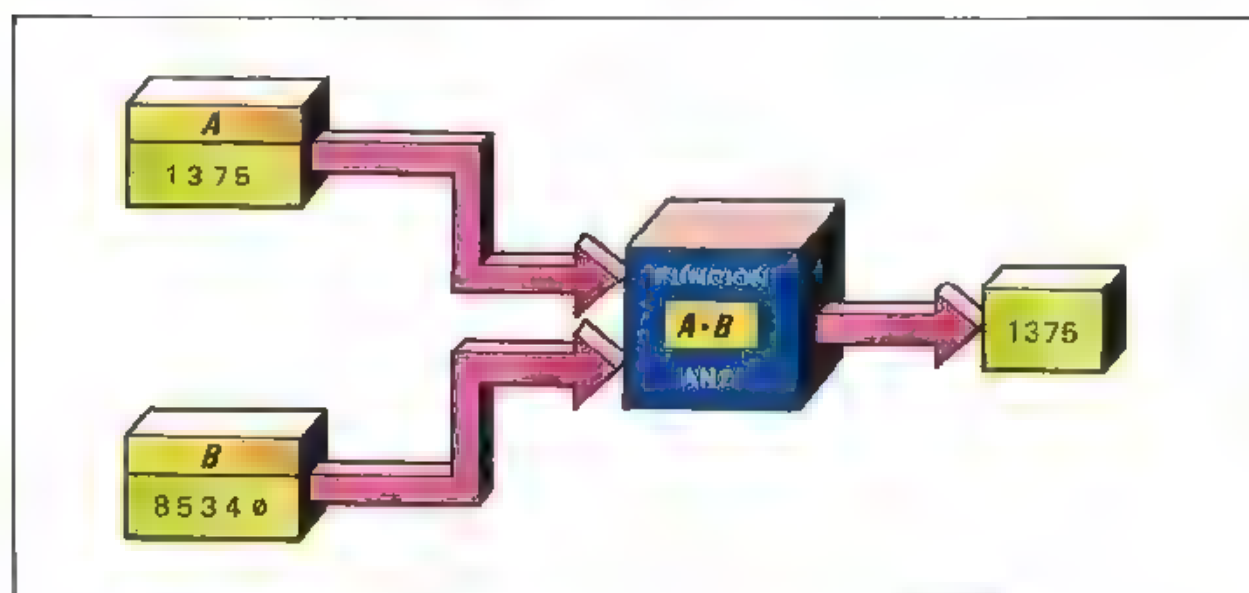
c) $3 \text{ OR } 0$

Resultado = 3

Como ejercicio, intente re-
solver la siguiente función

— $\text{NOT } ((X \text{ OR } Y) \text{ AND } (\text{NOT } Z))$

Para los valores: $X = 8$, $Y = 10$ y $Z = 830$. Si desea com-
probar el resultado de su cálculo
introduzca como comando
directo la función anterior pre-
cedida de la sentencia PRINT.



Función «NOT».

CÓDIGO ASCII

Para representar un carácter en la pantalla del televisor o en la impresora, el ordenador utiliza el *código ASCII*; éste permite la transferencia de datos entre el ordenador y los dispositivos conectados a él (periféricos). Cada número, letra o símbolo tiene su representación en este código.

ASCII es la abreviatura, en inglés, de «American Standard Code for Information Interchange» que, traducido al idioma español, significa «Código normalizado Americano para intercambio de información».

El ASCII completo consta de 256 caracteres, cuyo código está comprendido entre 0 y 255.

Edita el programa número «1» que, una vez ejecutado, visualiza en pantalla los caracteres ASCII usados por el ZX Spectrum y comprendidos entre el código 32 y el 255.

Manejo de la tabla

Para conocer el *código decimal* correspondiente a un determinado carácter ASCII, basta con sumar los números de *fila* y *columna* indicados en la tabla.

código = fila + columna

Ejemplos:

CARACTER ASCII	NUMERO		CODIGO
	FILE	COLUMNA	
£	30	5	35
USR	190	2	192
M	70	7	77
8	50	6	56

PROGRAMA 1

```

10 REM *****
   *
   *      CURSO
   *
   * BASIC/SINCLAIR
   *
   *      "ASCII"
   *
   * *****

20 BORDER 1: PAPER 1: INK 7: CLS

30 REM *****
   *
   * IMPRESION DE
   *
   * CARACTERES
   *
   * *****

40 FOR Y=32 TO 255 STEP 44
50 FOR X=Y TO Y+43
60 IF X<100 THEN PRINT " ",
70 PRINT X;" ";CHR$ X
80 IF X=255 THEN GO TO 200
90 NEXT X
100 PRINT #0,AT 1,0,"Pulse una
tecla para continuar."
110 PAUSE 0: BEEP 0.05,20
115 CLS
120 NEXT Y
200 REM

*****
*
* CONTINUACION ?
*
*****

210 POKE 23658,8
220 PRINT #0,AT 0,0;"Desea obtener un nuevo listado (S/N)"
230 PAUSE 0: LET D$=INKEY$
240 IF D$="S" THEN BEEP 0.05,20
GO TO 10
250 IF D$="N" THEN BEEP 0.05,20
: CLS : STOP
260 BEEP 0.2,-15: GO TO 230

```

El programa «2» visualiza el código decimal correspondiente a la cadena ASCII introducida, de un máximo de 20 caracteres.

Organización del ASCII

Dentro del código ASCII, usado por el Spectrum, pueden encontrarse diversas zo-

nas, teniendo cada una de ellas unas características distintas:

- CODIGO TRANSPARENTE.
- CODIGO ASCII CONVENCIONAL.
- CODIGO ASCII ESPECIFICO.
- GRAFICOS PREDEFINIDOS.

- GRAFICOS DE USUARIO.
- TOKENS.

El denominado *código transparente* está formado por una serie de comandos y funciones de control, como por ejemplo, el control de los cursores, el del color, la función EDIT, etc. Estos caracteres ASCII de control son específicos

OPERACION	SIMBOLO	PRIORIDAD	EJEMPLO
FRAGMENTACION	TO	12	"Juan" (1 TO 2)
POTENCIACION	↑	10	10 2
NEGACION	—	9	— 15
MULTIPLICACION	*	8	7 * 13
DIVISION	/		152 / 2
ADICION	+	6	10 + 4
SUBSTRACCION	-		8 - 5
OPERADORES RELACIONALES	=	5	10 4
	>		
	<		
	< >		
OPERADORES LOGICOS	> =	4	NOT 5
	< =		
	NOT		
	AND		
	OR	3	10 AND 1
		2	7 OR 0

del Spectrum, otros ordenadores disponen de otro juego distinto.

El código transparente está comprendido entre el 0 y el 31, en decimal. Dentro de éste, existen caracteres que no son utilizados, cuando el Spectrum representa uno de estos códigos aparece una interrogación (?) en su lugar.

Los códigos comprendidos entre el 32 y el 127 forman el ASCII convencional de todo ordenador. En esta zona se encuentran los caracteres correspondientes a las 26 letras mayúsculas, a sus homólogas las minúsculas, a los diez dígitos (0 al 9), al carácter «espacio» y a una serie de símbolos (€, &, &...) y signos ortográficos (":", "!", "?...).

A pesar de ser la zona convencional del ASCII, el Spectrum tiene dos caracteres particulares, los correspondientes a la «Libra» (£) y al «Copyright» (©), códigos 96 y 127, respectivamente.

Los gráficos predefinidos, es decir, los símbolos que están dibujados sobre las teclas con los números «1» a «8», y los complementarios, tienen un código comprendido entre el 128 y el 143. A continuación se encuentran los códigos correspondientes a los «DGU» (Gráficos Definidos por el Usuario), que como ya se indicó en el capítulo primero, éstos están asignados a las teclas con las letras de «A» y la «U» y su código está comprendido entre el 144 y el 164.

Al final del juego de caracteres se encuentran los *tokens*, término que simboliza a las 88 palabras clave (RUN, STOP, CLEAR...) una vez codificadas, éstas forman la lista de comandos, sentencias y funciones. También se encuentran en esta zona tres caracte-

PROGRAMA 2

```

10 REM *****
   *
   *      CURSO
   *
   * BASIC/SINCLAIR
   *
   * ASCII/DECIMAL
   *
   * *****

20 BORDER 1: PAPER 1: INK 7: C
L5
22 LET FLAG52=23658
24 LET PIP=23609
30 REM

   *****
   *
   *      ENTRADA DE
   *
   *      CARACTERES
   *
   * *****

32 POKE PIP,100
34 POKE FLAG52,0
40 INPUT "CADENA: ";A$
50 IF LEN A$>20 THEN GO TO 40
60 POKE PIP,0
70 REM

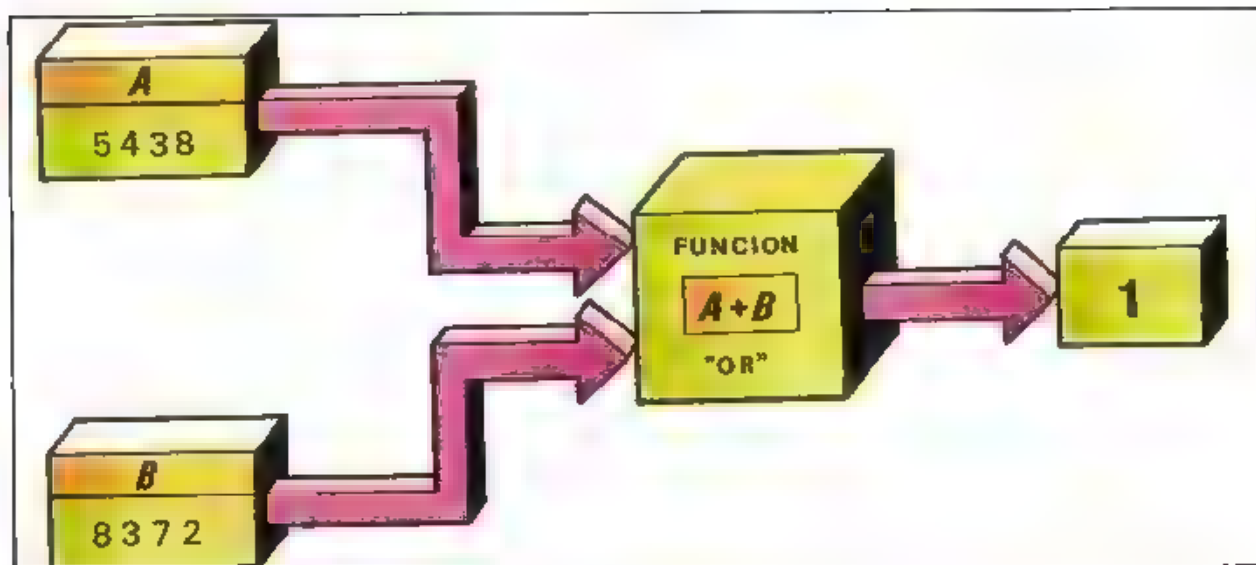
   *****
   *
   *      VISUALIZACION
   *
   * *****

80 FOR X=1 TO LEN A$
90 PRINT A$(X); " ", CODE A$(X)
100 NEXT X
110 REM

   *****
   *
   *      CONTINUACION
   *
   * *****

120 POKE FLAG52,8
130 PRINT #0;AT 1,0;"Quiere vol
ver a empezar (S/N) "
140 PAUSE 0: LET D$=INKEY$
150 IF D$="S" THEN BEEP 0.05,20
: GO TO 10
160 IF D$="N" THEN BEEP 0.05,20
: CLS : STOP
170 BEEP 0.2,-15: GO TO 140

```

terres que ocupan dos posiciones cada uno, estos son los correspondientes a los símbolos "<=", ">=", y "<>".

Transmisión del ASCII

En el lenguaje BASIC del Spectrum, la transmisión de caracteres ASCII puede efectuarse o directamente en este código o anteponiendo la sentencia "CHR \$" al correspondiente código decimal.

Ejemplos:

ASCII	DECIMAL
PRINT «A»	PRINT CHR\$ 65
PRINT «a»	PRINT CHR\$ 97

El primer método ofrece ciertas ventajas con respecto al segundo.

- Es legible directamente.
- Ocupa menos sentencias.
- Más rápido de ejecución.

si por el contrario todo o que desea es semi-camufiar el mensaje, será conveniente utilizar el segundo.

Ejecute la siguiente instrucción directa y compare el resultado con el proporcionado por el programa número «3».

PRINT «Maese Pérez
el organista»

PROGRAMA 3

```

10 REM *****
   *          CURSO          *
   * BASIC/SINCLAIR         *
   *    "CHR$"              *
   *          *****
20 BORDER 7: PAPER 7: INK 0: C
LS
30 REM *****
   * IMPRESION DE          *
   * CARACTERES            *
   *          *****

35 RESTORE
40 FOR X=1 TO 24
50 READ código decimal
60 PRINT CHR$ código decimal;
70 NEXT X
75 PRINT
80 REM *****
   *          TABLA DE      *
   * CARACTERES             *
   *          *****

90 DATA 77,97,101,115,101,32,8
0,101,114,101,122,32,101,106,32,
111,114,103,97,110,105,115,116,9
7

```

ORGANIZACION DEL JUEGO DE CARACTERES ASCII DEL SPECTRUM

ORGANIZACION DEL JUEGO DE CARACTERES "ASCII" USADO EN EL ZX SPECTRUM


COLUMNAS										
0	1	2	3	4	5	6	7	8	9	
0							PRINT coma	EDIT	cursor ladr.	cursor delu.
10	CURSOR abajo	cursor arriba	DELETE	ENTER	numero		INK control	PAPER control	FLASH control	BRIGHT control
20	INVERSE control	OVER control	AT control	TAB control						
30			espacio	!	"	#	\$	%	&	'
40	()	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[\]	^	-	ε	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z		!	!	~	⊙	□	■
130	▣	▤	▥	▦	▧	▨	▩	▪	▫	▬
140	▭	▮	▯	▰	(A)	(B)	(C)	(D)	(E)	(F)
150	(G)	(H)	(I)	(J)	(K)	(L)	(M)	(N)	(O)	(P)
160	(Q)	(R)	(S)	(T)	(U)	RND	INKEYS	PI	FN	POINT
170	SCREENS	ATTR	AT	TAB	VALS	CODE	VAL	JEN	SH	COS
180	DIM	ASH	ACS	ATN	LN	EXP	INT	SEN	SGN	ARS
190	PEEK	N	USR	STRG	CHR\$	NOT	SH	OR	AND	<=
200	>=	< >	LINE	THEN	TO	STEP	DEF FN	CAT	FORMAT	MOVE
210	ERASE	OPEN =	CLOSE =	MERGE	VERIFY	BEEP	CIRCLE	INK	PAPER	FLASH
220	BRIGHT	INVERSE	OVER	OUT	PRINT	LIST	STOP	READ	DATA	RESTORE
230	NEW	BORDER	CONT	DIM	REM	FOR	GOTO	GO SUB	INPUT	LOAD
240	LIST	LET	PAUSE	NEXT	POKE	PRINT	PLOT	RUN	SAVE	RAND
250	IF	CLS	DRAW	CLEAR	RETURN	COPY				

 CODIGO
TRANSPARENTE
ESPECIFICO.


 CARACTERES
"ASCII"
ESPECIFICOS.

 GRAFICOS
DE USUARIO.

 CODIGO NO
UTILIZADO.

 CARACTERES
"ASCII"
CONVENCIONAL.

 GRAFICOS
PREDEFINIDOS.

 JUEGO DE COMANDOS
SENTENCIAS
Y FUNCIONES.

OPERACIONES CON CADENAS

Concatenación de cadenas

Con este nombre se designa la operación de *encadenamiento* de un conjunto de constantes o variables alfanuméricas. La concatenación se realiza con el operador aritmético de la suma "+". Pueden tener diversas estructuras.

"Cons1" + "Cons2" + ... + "ConsN"
 Var 1\$ + Var2\$ + ... + VarN\$
 'Cons1' + 'Var1\$' + 'Var2\$' + ... + 'ConsN'

Cons = Constante de cadena

Var = Variable de cadena

Ejemplos:

— El resultado de la concatenación de las constantes "Esta frase" + "e encadenada" + "na" + "da" + " " + "es muy larga" es la siguiente:

"Esta frase encadenada es muy larga"

— Si asignamos a la variable a\$ el valor "ZX SP" y a b\$ el valor "ECTRUM", efectuando la operación c\$ = a\$ + b\$, la variable c\$ tendrá el valor:

"ZX SPECTRUM"

— Siendo a\$ = "Programas" y b\$ = "aplicacion", el resultado de a\$ + "de" + b\$ será:

"Programas de aplicación"

Como ejercicio demostrativo edite el programa número «1». Cuando se ejecuta, el ordenador espera que le sean introducidas tres cadenas, con una longitud máxima de 10 caracteres cada una. Posterior-

PROGRAMA 1

```

10 REM *****
   *
   *      CURSO
   *  BASIC/SINCLAIR
   *
   * *****
   *
   *  CONCATENACION
   *
   * *****

20 BORDER 1: PAPER 1 INK 7: C
LS
30 REM *****
   *
   *  INTRODUCCION
   *
   *  DE CADENAS
   *
   * *****

100 INPUT "CADENA 1: ",A$
110 IF LEN A$>10 THEN GO TO 100
115 PRINT "CADENA 1: ",A$, ""

120 INPUT "CADENA 2: ",B$
130 IF LEN B$>10 THEN GO TO 120
135 PRINT "CADENA 2: ",B$, ""

140 INPUT "CADENA 3: ",C$
150 IF LEN C$>10 THEN GO TO 140
155 PRINT "CADENA 3: ",C$, ""

160 PRINT "":A$+B$+C$;""
170 PRINT "":A$+C$+B$;""
180 PRINT "":B$+A$+C$;""
190 PRINT "":B$+C$+A$;""
200 PRINT "":C$+A$+B$;""
210 PRINT "":C$+B$+A$;""

220 REM *****
   *
   *  CONTINUACION?
   *
   * *****

230 POKE 23658,8
240 PRINT AT 21,0;"QUIERE CONTI
NUAR (S/N)"
250 PAUSE 0
260 LET D$=INKEY$
270 IF D$="S" THEN CLS . GO TO
100
280 IF D$="N" THEN CLS STOP
290 GO TO 250

```


mente visualiza en la pantalla todas las posibles combinaciones de concatenación. Introduzca, por ejemplo, su nombre y dos apellidos, como parámetros y observe el resultado.

Subcadena

Con el término de *subcadena* se designa a un grupo de caracteres consecutivos extraídos de una cadena. Por ejemplo 'dicc' es una subcadena de 'diccionario', también lo son 'onar' y 'ario'; sin embargo 'dic', 'noar' y 'aroi' no lo son, ya que sus caracteres no han sido extraídos consecutivamente.

Otros ejemplos son:

CADENAS	SUBCADENAS
"Coche de carreras "	"che de"
"Ángel, Pepe, Luis"	"Pepe"
"Lunesmartesmiércoles"	"lunes"
"Ordenador personal"	"Ordena"

Fragmentación

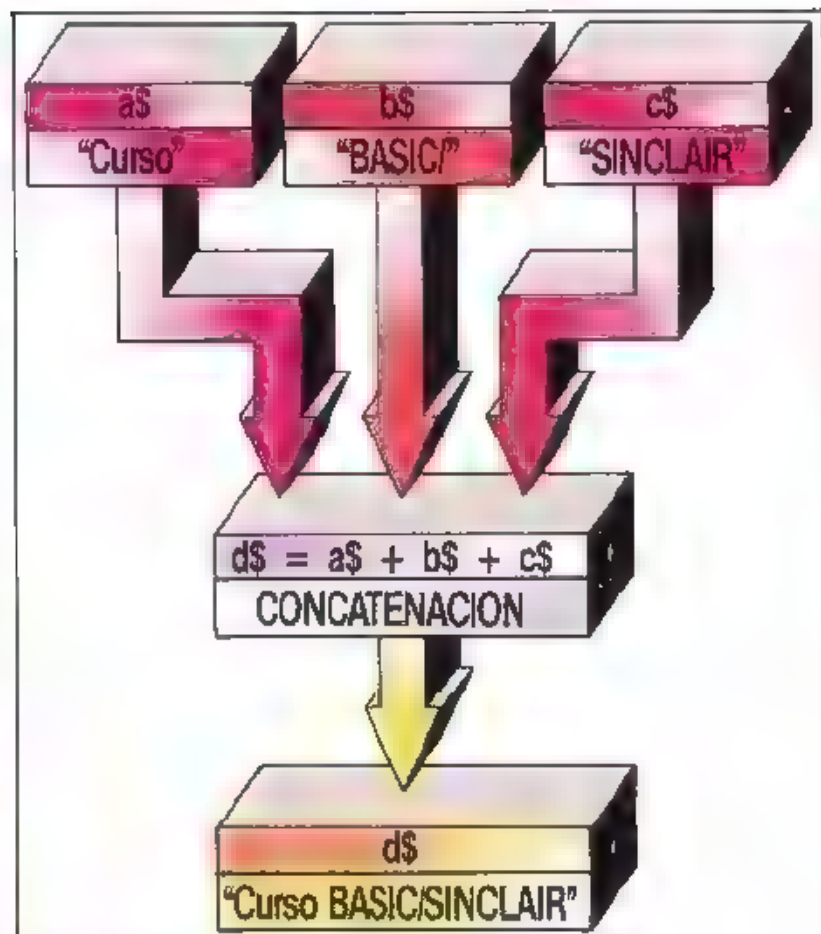
Para extraer una subcadena se utiliza la operación de *fragmentación*. La estructura general es:

FRAGMENTACION
cadena (a TO b)

donde «a» es el número del primer carácter a extraer y «b» el último, entre ambos limitan la longitud de la subcadena, «a» y «b» pueden ser constantes o variables numéricas, pero siempre positivas, de lo contrario, el ordenador enviará el mensaje:

B Integer out of range

La cadena puede ser o bien



Concatenación.

una constante o bien una variable.

Ejemplos.

FRAGMENTACION	RESULTADO
"Curso BASIC" (1 TO 5)	"Curso"
"Monitor TV" (5 TO 9)	"tor T"
"MICROHOBBY" (6 TO 10)	"HOBBY"
"Cartucho" (4 TO 5)	"tu"

Si «b» fuera menor que «a», el resultado de la fragmentación sería una cadena vacía.

Ejemplos:

FRAGMENTACION	RESULTADO
"Juan" (2 TO 1)	Cadena vacía ("")
"Pepe" (20 TO 10)	Cadena vacía ("")

Cuando «a» es 0 o «b» es mayor que el correspondiente a la longitud de la cadena, el

ordenador envía el siguiente mensaje:

3 Subscript wrong

Ejemplo.

- "Televisor" (0 TO 5)
- "Teclado" (2 TO 30)

El programa número "2" permite la introducción de una cadena de quince caracteres como máximo, es necesario además introducir los límites de la fragmentación. La subcadena resultante es visualizada en la pantalla.

Fragmentación específica

Existen tres tipos de fragmentación que tienen una estructura distinta:

- Fragmentación de un solo carácter.

- Fragmentación izquierda.
- Fragmentación derecha.

Cuando se quiere extraer un solo carácter de una cadena, la estructura es la siguiente:

FRAGMENTACION
cadena (n)

donde «n» es el número de carácter; es equivalente a escribir «cadena» (n TO n).

Ejemplos:

FRAGMENTACION	RESULTADO
"Cassette" (5)	"e"
"Cable" (2)	"a"
"Circular" (3)	"y"
"Cigarrillos" (8)	"l"

La fragmentación izquierda consiste en la extracción de los «n» primeros caracteres de una cadena, su estructura es:

FRAGMENTACION
cadena (TO n)

equivale a «cadena» (1 TO n)

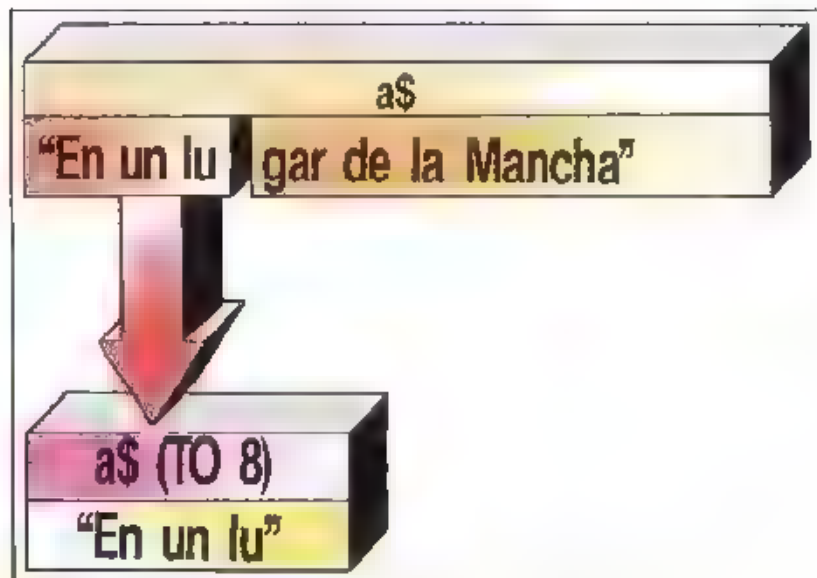
Ejemplos:

FRAGMENTACION	RESULTADO
"CARAMELO" (TO 3)	"CAR"
"coche" (TO 2)	"co"
"Botella" (TO 5)	"Botel"
"Corazon" (TO 4)	"Cora"

La fragmentación derecha permite extraer los últimos caracteres de una cadena a partir de «n», su estructura es la siguiente:

FRAGMENTACION
cadena (n TO)

es equivalente a «cadena» (n TO fin).



Fragmentación izquierda.

PROGRAMA 2

```

10 REM *****
   *          CURSO          *
   * BASIC/SINCLAIR        *
   * *****              *
   * FRAGMENTACION        *
   * *****              *

20 BORDER 1: PAPER 1 INK 7: C
LS
25 POKE 23658,0
30 REM *****
   * INTRODUCCION          *
   * CADENA                 *
   * *****              *

100 INPUT "CADENA ",A$
105 IF A$="" THEN GO TO 100
110 IF LEN A$>15 THEN GO TO 100
115 PRINT "CADENA ";A$;" "
120 LET longitud=LEN (A$)
125 PRINT "LONGITUD. ",longitud
130 REM *****
   * INTRODUCCION          *
   * LIMITES                *
   * *****              *

140 PRINT AT 21,0;"LIMITE INFER
ION \1 ",longitud," "
145 INPUT LINE I$
150 IF I$<"1" OR I$>"9" THEN GO
TO 145

```

```

160 IF VAL (I$)>longitud THEN G
O TO 145
162 LET inferior=VAL (I$)
164 PRINT AT 4,0;"LIMITE INFERI
OR inferior
170 PRINT AT 21,7, SUP"
175 INPUT LINE S$
180 IF S$<"1" OR S$>"9" THEN GO
TO 175
190 IF VAL (S$)>longitud THEN G
O TO 175
192 LET superior=VAL (S$)
194 PRINT AT 5,0;"LIMITE SUPERI
OR superior
196 PRINT AT 21,0;";

200 REM *****
*      *
*  VISUALIZACION  *
*      *
*  SUBCADENA      *
*      *
*****

210 PRINT AT 8,0;"SUBCADENA. "
";A$(inferior TO superior);""
220 REM *****
*      *
* CONTINUACION?  *
*      *
*****

225 POKE 23658,8
230 PRINT AT 21,0,"QUIERE CONTI
NUAR (S/N)"
240 PAUSE 0
250 LET D$=INKEY$
260 IF D$="S" THEN CLS : GO TO
100
270 IF D$="N" THEN CLS : STOP
280 GO TO 240

```

Ejemplos:

FRAGMENTACION	RESULTADO
"Zumos" (3 TO)	"mos"
"pescados" (5 TO)	"ados"
"Puros" (2 TO)	"uros"
"A12557" (4 TO)	"557"

Asignación de subcadenas

Una variable de cadena puede ser modificada parcialmente utilizando la asignación de subcadenas. Supongamos que la variable a\$ tiene asignado un valor.

"Evolucion de los anfibios" utilizando la expresión:

a\$ (18 TO 25) = "caballos"
la variable a\$ tendrá un nuevo valor

"Evolucion de los caballos"

Cuando a una subcadena se le asigna una longitud de caracteres mayor que los expresados, automáticamente los caracteres sobrantes, de la derecha, son recortados.

Ejemplo:

a\$ = "*****"

asignando

a\$ (2 TO 5) = "\$\$\$\$\$"

la variable a\$ tendrá el valor

"\$\$\$\$*****"

Cuando por el contrario la longitud sea inferior, a subcadena se rellena con espacios.

Ejemplo:

a\$ = "*****"

asignando

a\$ (2 TO 5) = "\$\$"

a\$ tendrá el valor

"\$\$ *****"

Este método de recortar o rellenar con espacios se llama «asignación procusteana». Utilizando este método se puede asignar un nuevo valor a toda la variable manteniendo la longitud de ésta.

Ejemplo:

a\$ = "*****"

asignando

a\$ () = "\$\$\$"

el nuevo valor de a\$ será

"\$\$\$"

Comparación de cadenas

Las cadenas al igual que los números pueden compararse haciendo uso de los operadores relacionales. Para averiguar si una cadena es mayor, menor o igual a otra, se comparan los códigos "ASCII" correspondientes al primer carácter de cada una de ellas; si estos son diferentes, será mayor la que tenga un código más alto.

Ejemplo 1:

a\$
P A L O M A
↓
G O R R I O N
b\$

LETRA	ASCII
P	80
G	71

P < G

luego a\$ > b\$

Ejemplo 2:

a\$
B L T H E
↓
D U I E
b\$

LETRA	ASCII
B	66
b	98

B < b

luego a\$ < b\$

Si fueran iguales los códigos del primer carácter, sería necesario pasar a comparar los siguientes hasta encontrar uno diferente.

Ejemplo



LETRA	ASCII
J	74
T	84

J < T

luego a\$ < b\$

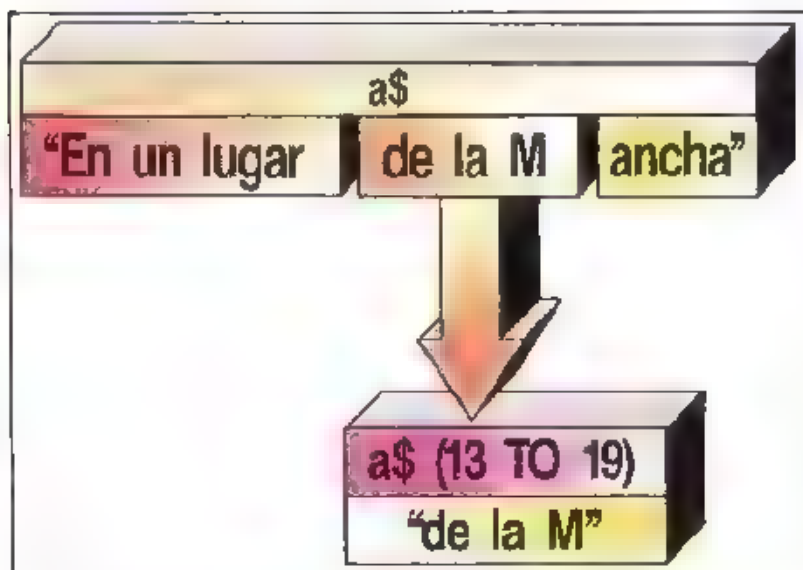
En este proceso de comparación puede ocurrir que los caracteres de una cadena se acaben antes que los de otra, en este caso, es mayor la que tenga mayor longitud.

Ejemplo:



luego a\$ > b\$

Si las dos cadenas se acaban sin encontrar ningún carácter distinto, significa que ambas son iguales.



Fragmentación central.

Ejemplo.

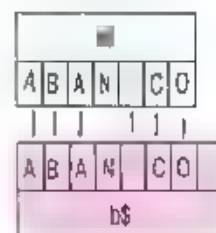


luego a\$ = b\$

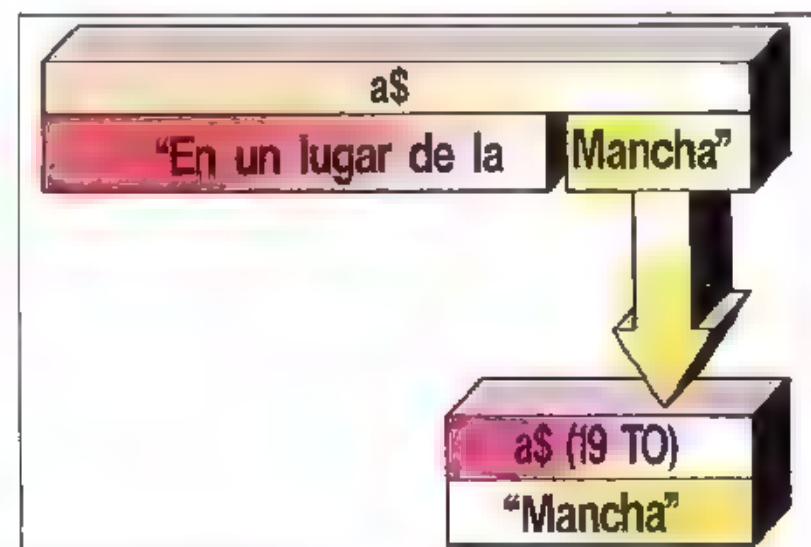
El carácter «espacio» también tiene su correspondiente código ASCII, por este motivo, al comparar cadenas alfanuméricas debe ser tenido en

cuenta. Aparentemente, en el siguiente ejemplo, las dos cadenas son iguales, sin embargo no lo son, ya que una de ellas contiene al final un espacio.

Ejemplo:



luego a\$ < b\$



Fragmentación derecha.

PROGRAMA 3

```

10 REM *****
  *
  *      CURSO
  *
  *  BASIC/SINCLAIR
  *
  *  "COMPARACION"
  *
  *****

20 BORDER 1: PAPER 1: INK 7: C
LS
50 REM *****
  *
  *  ENTRADA DE DATOS
  *
  *****

100 INPUT "CADENA 1: ", A$
110 IF LEN A$ > 10 THEN GO TO 100
115 PRINT "CADENA 1: ", A$, ""

120 INPUT "CADENA 2: ", B$
130 IF LEN B$ > 10 THEN GO TO 120
135 PRINT "CADENA 2: ", B$, ""

138 REM *****
  *
  *  COMPARACION
  *
  *****

140 IF A$ > B$ THEN PRINT " ", A$
    > " ", B$, ""
150 IF A$ < B$ THEN PRINT " ", A$
    < " ", B$, ""
160 IF A$ = B$ THEN PRINT " ", A$
    = " ", B$, ""
200 REM *****
  *
  *  CONTINUACION?
  *
  *****

210 POKE 23658,8
220 PRINT AT 21,0;"QUIERE CONTI
NUA
(S/N) "
230 PAUSE 0
240 LET D$=INKEY$
250 IF D$="S" THEN CLS : GO TO
100
260 IF D$="N" THEN CLS : STOP
270 GO TO 230

```

El programa número «3» compara dos cadenas cualesquiera, introducidas por el teclado, de un máximo de diez caracteres cada una, y visualiza en la pantalla el resultado de la comparación.

Ordenación de cadenas

Comparar cadenas puede tener utilidad, por ejemplo, en

la ordenación por orden alfabético de un fichero de nombres. Por sucesivas comparaciones de los elementos del fichero se puede llegar a a ordenación de las cadenas.

Veamos por pasos cómo se ordena un fichero que contiene los siguientes datos.

"PEPE" "JUAN" "ANTONIO",
"RODRIGO" y "DAVID"

como se puede observar, desordenados alfabéticamente.

PRIMER PASO

El primer dato leído es "PEPE", este lo colocamos provisionalmente en la primera posición.

SEGUNDO PASO:

El dato "JUAN" se compara con "PEPE".

"JUAN" < "PEPE"

"PEPE" se transfiere a la segunda posición

TERCER PASO:

"ANTONIO" es el tercer dato leído y se compara con "JUAN" y "PEPE".

"ANTONIO" < "JUAN" < "PEPE"

Las cadenas "JUAN" y "PEPE" se desplazan para que "ANTONIO" ocupe la primera posición.

CUARTO PASO:

El código correspondiente a "RODRIGO" es superior al de los demás.

"RODRIGO" > "PEPE" >
"JUAN" > "ANTONIO"

"RODRIGO" se coloca en la cuarta posición, a continuación de los demás.

QUINTO PASO:

El último dato leído es "DAVID" y tiene un código superior al de "ANTONIO" e inferior al resto.

"DAVID" > "ANTONIO"

"DAVID" < "JUAN" < "PEPE" < "RODRIGO"

"JUAN", "PEPE" y "RODRIGO" tienen que desplazarse para que "DAVID" ocupe la segunda posición


```

*****
260 FOR X=2 TO orden
270 INPUT "CADENA ";D$
280 IF LEN (D$)>longitud THEN G
O TO 270
290 REM
*****
*
* RELLENO ESPACIOS *
*
*****
300 FOR Y=1 TO longitud-LEN (D$
)
310 LET D$=D$+" "
320 NEXT Y
325 REM
*****
*
* COMPARA CADENAS *
*
*****
330 IF D$(N$ X-1) THEN GO SUB 1
000 GO TO 360
340 LET N$(X)=D$
350 PRINT AT X,11,D$
360 NEXT X
370 REM
*****
*
* CONTINUACION? *
*
*****
380 POKE 23658,8
390 PRINT #0,AT 0,0;"QUIERE CON
TINUAR (S/N)";
400 PAUSE 0
410 LET D$=INKEY$
420 IF D$="S" THEN GO TO 500
430 IF D$="N" THEN CLS STOP
440 GO TO 400

```

```

500 REM
*****
*
* BORRADO CADENAS *
*
*****
510 FOR Y=1 TO 20
520 PRINT AT Y,11,"
"
525 LET N$(Y)=" "
530 NEXT Y
540 GO TO 210
1000 REM
*****
*
* SUB. DETECCION *
*
*****
1010 FOR Z=X-1 TO 1 STEP -1
1020 IF D$(N$(Z)) THEN GO TO 1050
1030 NEXT Z
1040 REM
*****
*
* ORDENA *
*
*****
1050 FOR Y=orden 1 TO Z+1 STEP -
1
1060 LET N$(Y+1)=N$(Y)
1070 PRINT AT Y+1 11,N$(Y)
1080 NEXT Y
1090 REM
*****
*
* INSERTA *
*
*****
1100 LET N$(Y+1)=D$
1110 PRINT AT Y+1,11,D$
1120 RETURN

```

Elaboración de programas

Antes de introducir las instrucciones necesarias para elaborar un programa, es necesario haber efectuado antes una serie de pasos. Existen diversas técnicas para ayudar al diseñador de programas a estructurar en alguna medida su trabajo y así poder conseguir unos resultados satisfactorios.

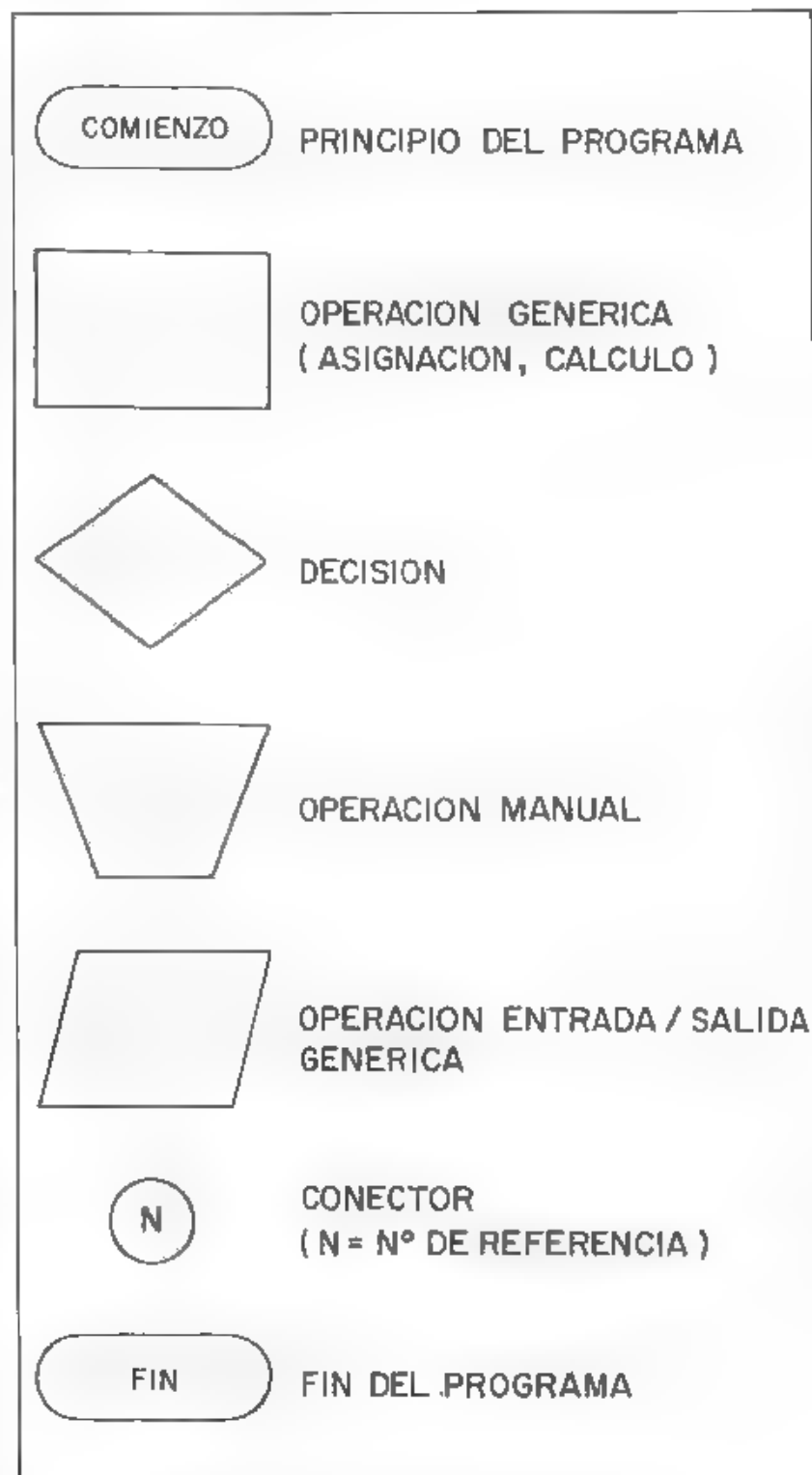
Básicamente todas las técnicas tienden a examinar globalmente el programa desde una perspectiva superior, e ir descendiendo, poco a poco, hasta llegar a elaborarlo por completo. En este proceso se pueden distinguir tres fases:

- Análisis del programa
- Síntesis.
- Representación gráfica.

Análisis

El primer paso es analizar el problema y determinar las funciones de nuestro programa, es decir, lo que deseamos que haga. Una vez conocido esto, podremos averiguar cuáles son los datos que debemos proporcionar al programa. Veamos un ejemplo.

Supongamos que deseamos confeccionar un programa que visualice en pantalla o nos saque por impresora un listado ordenado de nuestra agenda telefónica, ¿cuáles van a ser los datos que vamos a necesitar?, en principio los «nombres» y los «teléfonos» de nuestras amistades, con esta información se confecciona un boceto similar al de la figura.



Simbología utilizada en los diagramas de flujo.

En un paso siguiente podremos determinar los programas que vamos a necesitar; siguiendo con nuestro ejemplo necesitaremos.

- Un programa o rutina que nos permita la introducción de los datos por primera vez.
- Otro para poder introducir o modificar nuevos teléfonos.
- Un programa que ordene por orden alfabético los nombres y los visualice.
- Y ya por último, un programa que nos permita seleccionar cualquiera de los anteriores.

Síntesis

Las funciones de cada uno de los programas mencionados anteriormente, pueden ser esquematizados usando la técnica desarrollada por la compañía IBM y denominada HIPO, en ciales de «Hierarchy Input/Process/Output», que traducido al español significa Jerarquía Entrada/Proceso/Salida. Esta técnica consiste en definir un programa mediante tres bloques principales.

Entrada: definición de los datos a utilizar

Proceso: descripción esquematizada de los procesos o cálculos a realizar

- Salida: especificación de los datos a imprimir o visualizar.

En fases posteriores cada bloque se desarrolla en otros del mismo tipo, más completos, pero todavía sin llegar al detalle.

Representación gráfica

Es el último paso antes de la edición de un programa. Es-

PROGRAMA 1

```

10 REM *****
   *
   *      CURSO      *
   *
   * BASIC/SINCLAIR  *
   *
   *      LISTIN      *
   *
   *****

12 BORDER 2: PAPER 2: INK 6: C
LS
14 PRINT FLASH 1, AT 7, 10; "PARE
  LA CINTA"
16 PLOT 104, 56: DRAW 16, 31: DR
  AW 15, 0: DRAW 16, -31: DRAW -47, 0
18 CIRCLE 128, 71, 7
20 PLOT 96, 80: DRAW 15, 0: DRAW
  7, 11: DRAW 19, 0: DRAW 7, -11: DR
  AW 15, 0: DRAW -17, 17: DRAW -30, 0
  : DRAW -17, -17
22 PLOT 144, 72: DRAW 24, 0: DRA
  W 0, -41: DRAW -80, 0
24 PRINT INVERSE 1; AT 16, 13, "L
  ISTIN"
26 PLOT 0, 0: DRAW 0, 175: DRAW
  255, 0: DRAW 0, -175: DRAW -255, 0
28 PLOT 0, 152: DRAW 255, 0
29 RESTORE
30 PRINT AT 1, 1; " "
31 FOR X=1 TO 28
32 READ chr
34 PRINT CHR$ chr;: BEEP 0.05,
  chr/2
36 NEXT X
38 DATA 127, 32, 77, 73, 67, 82, 79,
  72, 79, 66, 66, 89, 32, 38, 32, 82, 65, 70,
  , 65, 69, 76, 32, 80, 82, 65, 68, 69, 83
40 PRINT AT 7, 10; "
  "
42 PRINT #0; AT 1, 1; "Pulse una
  tecla para continuar"
44 PAUSE 0
46 BEEP 0.2, 20

100 REM *****
   *
   * DEFINICIONES *
   *
   *****

110 BORDER 1: PAPER 1: INK 7: C
LS
120 DIM N$(20, 22)
130 LET PIP=23609
140 LET FLAGS2=23658
145 LET datos=0
150 LET ordena=0
160 GO TO 500
200 REM

```



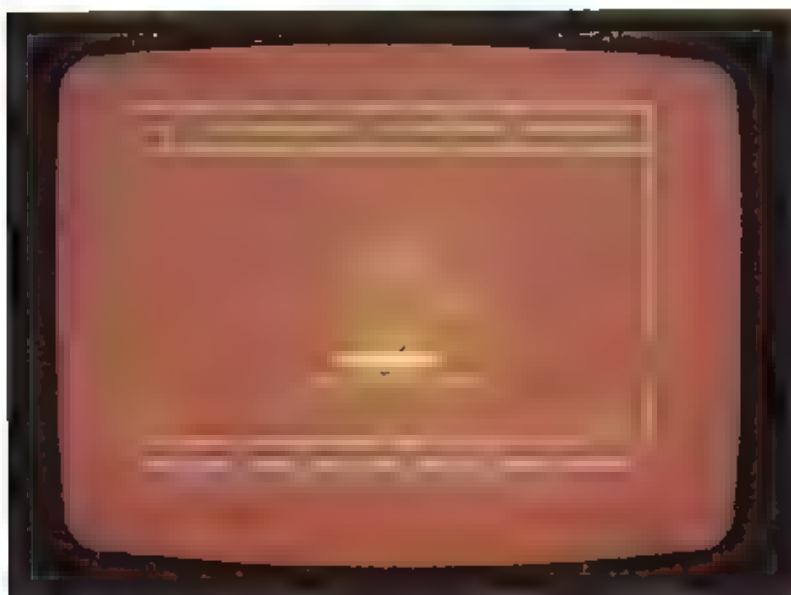
```

*****
*
*  NUEVO LISTIN  *
*
*****

205 CLS
208 DIM N$(20,22)
210 FOR Y=1 TO 20
220 GO SUB 800
230 IF D$='FIN' THEN GO TO 350
250 LET N$(Y,1 TO 15)=D$
260 PRINT D$,
270 GO SUB 920
320 LET N$(Y,16 TO 22)=D$
330 PRINT D$
340 NEXT Y
350 GO SUB 5000
355 CLS
360 GO SUB 5200
370 LET datos=1
375 LET ordena=0
380 GO TO 500
500 REM

*****
*
*  OPCIONES  *
*

```



Carátula del listin.

ta representación utiliza una serie de símbolos standard para facilitar su análisis por otras personas. El gráfico resultante se denomina «Diagrama de flujos», traducción de la palabra inglesa «Flow chart».

Los símbolos utilizados indican las operaciones que de-

be realizar el ordenador en cada momento.

Hay un símbolo que indica cuál es el comienzo o final de un programa. Otro representa a las operaciones genéricas, como por ejemplo, un cálculo, una asignación de valores, etc. El símbolo de *decisión* (rombo

invertido) realiza una comparación entre dos o más valores y, en función del resultado, elige una de las dos salidas posibles. La operación manual se refiere a toda tarea que el ordenador no pueda efectuar y que se necesite de una persona para realizarla, por ejemplo, poner en funcionamiento una impresora, introducir una cinta para su lectura o grabación, etc.

La operación genérica entrada/salida representa a cualquier función de lectura, grabación, impresión, etc., es decir, de entrada o salida de datos. Cuando el diagrama es bastante grande o complejo y se dibuja en varias hojas, la unión de éstas se realiza con el símbolo denominado *conector*, dentro de éste debe figurar un número para utilizarlo como referencia.

Dentro de los símbolos debe incluirse una descripción corta y clara de la operación cálculo, decisión, etc. que realizan. Los símbolos deben unirse mediante líneas con unas flechas que indiquen la dirección de ejecución del programa.

Programa

El programa "LISTIN" (número 4) realiza las funciones expresadas en los diagramas de flujo que acompañan este artículo. Una vez editado grábelo en cinta con el siguiente comando directo:

```
SAVE "LISTIN" LINE 10
```

de esta forma se autoejecuta a cargarlo.

Después de la presentación aparece un menú con diversas opciones. La número 1 permite crear un fichero con los

```

*****
502 POKE PIP,100
505 POKE FLAG32,0
506 CLS
510 PRINT AT 4,10,"OPCIONES"
520 PRINT AT 8,5,"1.- NUEVO LIS
TIN"
530 PRINT AT 12,5,"2.- MODIF. L
ISTIN"
540 PRINT AT 16,5,"3.- LISTIN"
545 PRINT AT 20,5,"4.- FIN"
550 PAUSE 0: LET P$=INKEY$
560 IF P$="1" THEN BEEP 0.05,20
GO TO 200
570 IF P$="2" THEN BEEP 0.05,20
GO TO 650
580 IF P$="3" THEN BEEP 0.05,20
GO TO 1000
585 IF P$="4" THEN BEEP 0.05,20
POKE PIP,A: GOTO STOP
590 BEEP 0.2,-15: GO TO 550

```

650 REM

```

*****
* MODIFICACION *
*
*****

```

```

660 IF datos=1 THEN GO TO 690
670 GO SUB 5500
680 LET datos=1
690 CLS
695 LET ordena=0
700 GO SUB 2000
740 FOR X=1 TO ultimo
745 IF X<10 THEN PRINT
750 PRINT X," ",N$(X,1 TO 15)
,N$(X,16 TO 22)
760 NEXT X
770 PRINT #0,AT 0,0,"M=Modifica
I=Inserta / F=Fin"
780 PAUSE 0: LET P$=INKEY$
782 IF P$="M" THEN BEEP 0.05,20
GO TO 800
784 IF P$="I" THEN BEEP 0.05,20
GO TO 830
786 IF P$="F" THEN BEEP 0.05,20
GO TO 890
788 BEEP 0.2,-15: GO TO 780
800 INPUT "Que linea ? ", LINE
P$
802 IF P$="" THEN GO TO 800
804 IF LEN P$>2 THEN BEEP 0.2,-
15: GO TO 800
806 FOR X=1 TO LEN P$
808 IF P$(X)<"0" OR P$(X)>"9" T
HEN BEEP 0.2,-15: GO TO 800
810 NEXT X
812 LET linea=VAL (P$)
814 IF P$<"1" OR linea>ultimo T
HEN BEEP 0.2,-15: GO TO 800
816 GO SUB 900
818 LET N$(linea,1 TO 15)=D$
820 PRINT AT linea-1,3 N$(linea
,1 TO 15);
822 GO SUB 920
824 LET N$(linea,15 TO 22)=D$
826 PRINT " ",N$(linea,15 TO 22)
)
828 GO TO 770
830 IF ultimo=20 THEN GO TO 850
831 LET ultimo=ultimo+1
832 PRINT #0,AT 0,0,"
834 GO SUB 900
836 LET N$(ultimo,1 TO 15)=D$
837 IF ultimo<10 THEN PRINT AT
ultimo-1,0," ",ultimo," " GO T
O 830
838 PRINT AT ultimo-1,0,ultimo,
839 PRINT N$(ultimo,1 TO 15);"
840 GO SUB 920
842 LET N$(ultimo,15 TO 22)=D$
844 PRINT N$(ultimo,15 TO 22)
846 GO TO 770

```

```

850 BEEP 0.2,-15
855 PRINT #0,AT 0,0,"No puede i
nserir mas nombres"
860 FOR T=1 TO 200 NEXT T
870 GO TO 770
880 PRINT #0,AT 0,0,"
891 GO SUB 5000
892 CLS
894 GO SUB 5200
896 GO TO 500
900 REM

```

```

*****
* ENTRADA NOMBRE *
*
*****

```

```

901 INPUT "NOMBRE " LINE D$
902 IF D$="" THEN GO TO 900
904 IF LEN D$>15 THEN BEEP 0.2,-
15: GO TO 900
906 FOR X=1 TO LEN D$
908 IF D$(X)<" " AND (D$(X)<"A
OR D$(X)>"Z") THEN BEEP 0.2,-1
5: GO TO 900
910 NEXT X
912 RETURN
920 REM

```

```

*****
* ENTRADA TELEF. *
*
*****

```

```

921 INPUT "TELEFONO " LINE D$
922 IF D$="" THEN GO TO 920
924 IF LEN D$>7 THEN BEEP 0.2,-
15: GO TO 920
926 FOR X=1 TO LEN D$
928 IF D$(X)<"0" OR D$(X)>"9" T
HEN BEEP 0.2,-15: GO TO 920
930 NEXT X
932 RETURN

```

1000 REM

```

*****
* LISTIN *
*
*****

```

```

1002 IF datos=1 THEN GO TO 1015
1010 GO SUB 5500
1012 LET datos=1
1015 CLS
1016 IF ordena=1 THEN GO TO 1050
1018 PRINT AT 10,0,"ESPERE UN MO
MENTO"
1020 GO SUB 2000
1030 DIM C$(20,22)
1032 LET C$(1,N$(1))
1034 FOR X=2 TO ultimo
1036 IF N$(X)<C$(X-1) THEN GO TO
1040
1038 LET C$(X)=N$(X): GO TO 1054
1040 FOR Z=X-1 TO 1 STEP -1
1042 IF N$(X)>C$(Z) THEN GO TO 1
046
1044 NEXT Z
1046 FOR Y=(ultimo-1) TO Z+1 STE
P -1
1048 LET C$(Y+1)=C$(Y)
1050 NEXT Y
1052 LET C$(Y+1)=N$(X)
1054 NEXT X
1056 CLS
1058 LET ordena=1
1060 FOR X=1 TO ultimo
1062 PRINT C$(X,1 TO 15);" ",C$(
X,16 TO 22)
1064 NEXT X
1100 GO SUB 5000
1110 GO TO 500
2000 REM

```

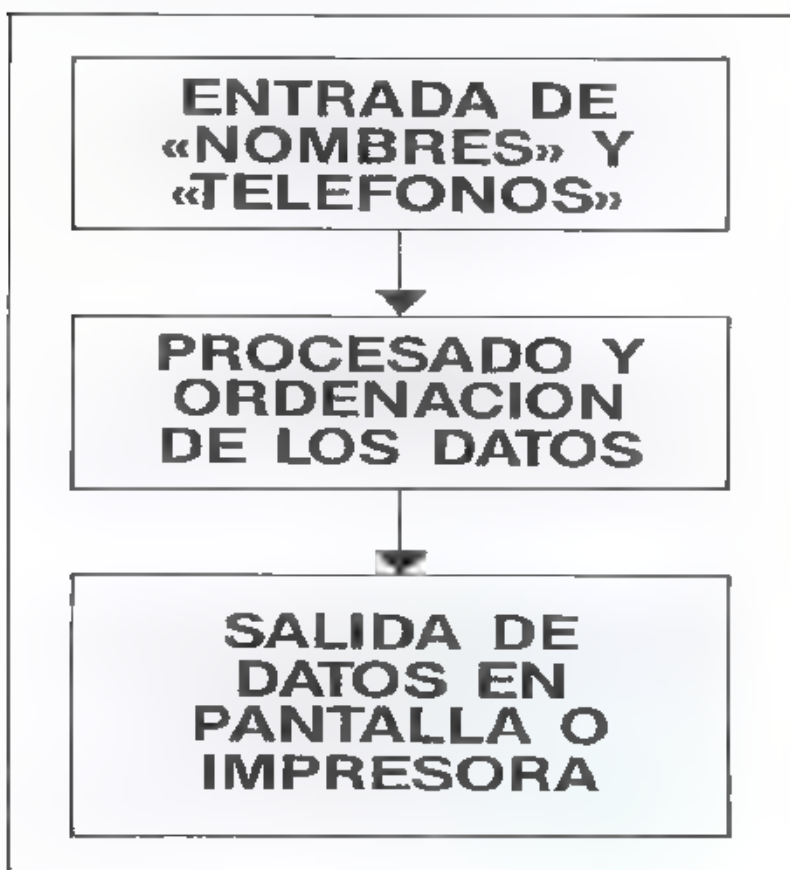
```

*****
* ULTIMO *
*
*****

```



Jerarquia «top-down» o descendente.



Fase de análisis.

nombres y teléfonos que deseamos almacenar

Una vez elegido, el programa nos pide la introducción del primer nombre, y a continuación el teléfono; recuerde que debe pulsar «ENTER» cada vez que introduzca un dato. El nombre debe constar como máximo de 15 caracteres, estos sólo pueden ser o letras mayúsculas o espacios, y el teléfono debe estar constituido como máximo por 7 cifras.

Una vez introducidos, el programa nos pedirá los siguientes y así hasta que se complete el listín, que puede estar formado por 20 nombres como máximo. Si no desea rellenar todo el listín, cuando no tenga más nombres para introducir teclee la palabra FIN.

Posteriormente se nos presenta la opción de sacar un listado por impresora y, a continuación, el programa da las instrucciones necesarias para grabar y verificar una cinta con los datos editados.

Con la opción 2 se puede modificar una cinta ya editada. Lo primero que hace es leer la cinta con los datos. A continuación se nos presenta otro menú con tres opciones, una para poder modificar un nombre ya existente, para lo cual nos pide el número de línea que deseamos modificar. Otra opción permite introducir nuevos nombres, y por último, la tercera imprime un listín y graba una cinta de la misma manera que la opción 1.

La opción 3 es la encargada de ordenar los nombres alfabéticamente y presentarlos en pantalla. Los datos son leídos de la cinta editada. También permite sacar un listado ordenado en impresora.

Para salir del programa seleccione la opción 4.

ENTRADA	PROCESO	SALIDA
INTRODUCCION DE DATOS POR TECLADO	ALMACENAMIENTO EN MEMORIA	GRABACION EN CINTA

EDITAR UN NUEVO «LISTIN TELEFONICO»

ENTRADA	PROCESO	SALIDA
LECTURA DE DATOS EN CINTA	MODIFICACION	NUEVA GRABACION

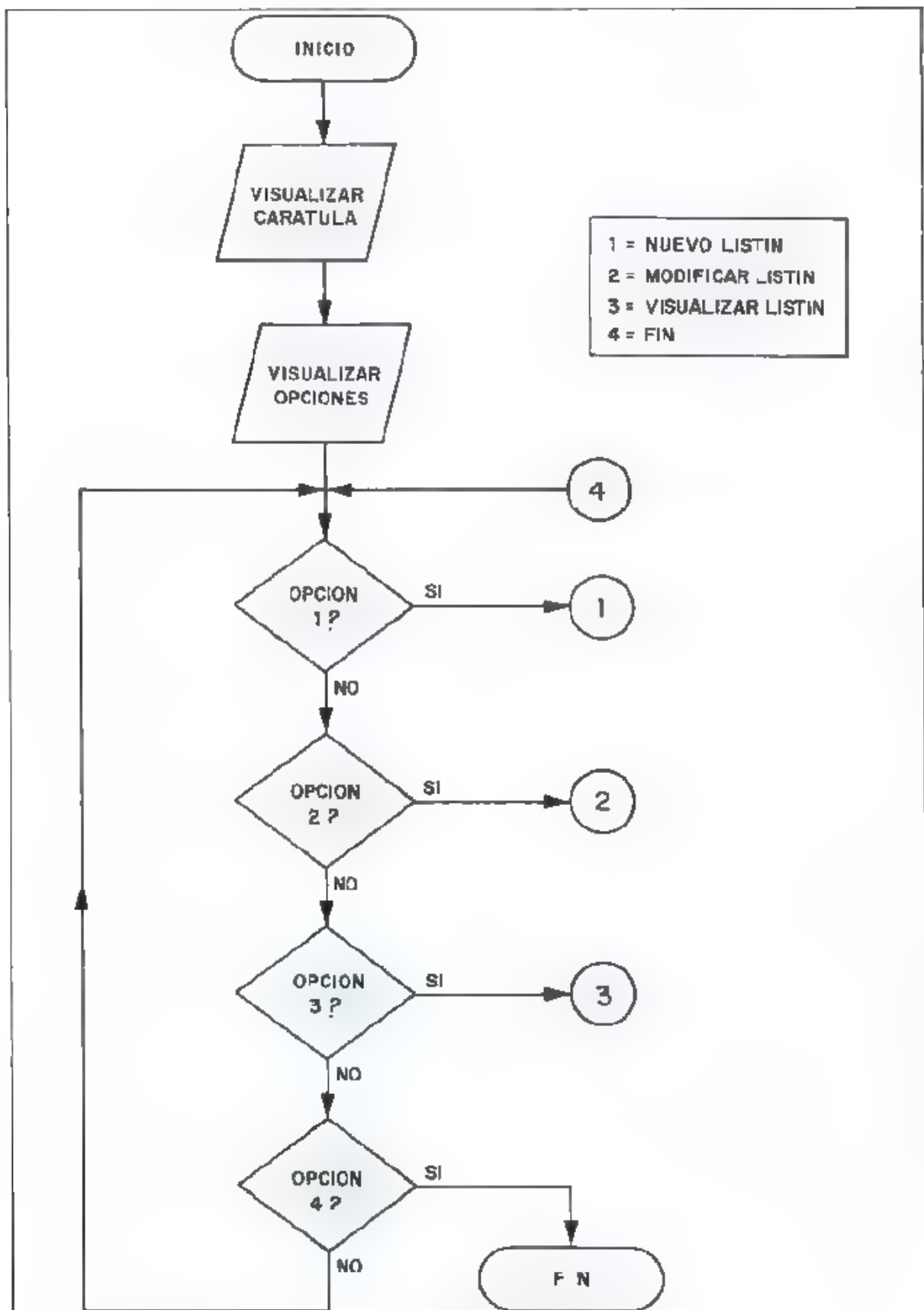
MODIFICACION DEL «LISTIN»

ENTRADA	PROCESO	SALIDA
LECTURA DE DATOS EN CINTA	ORDENACION DE DATOS	VISUALIZACION O IMPRESION

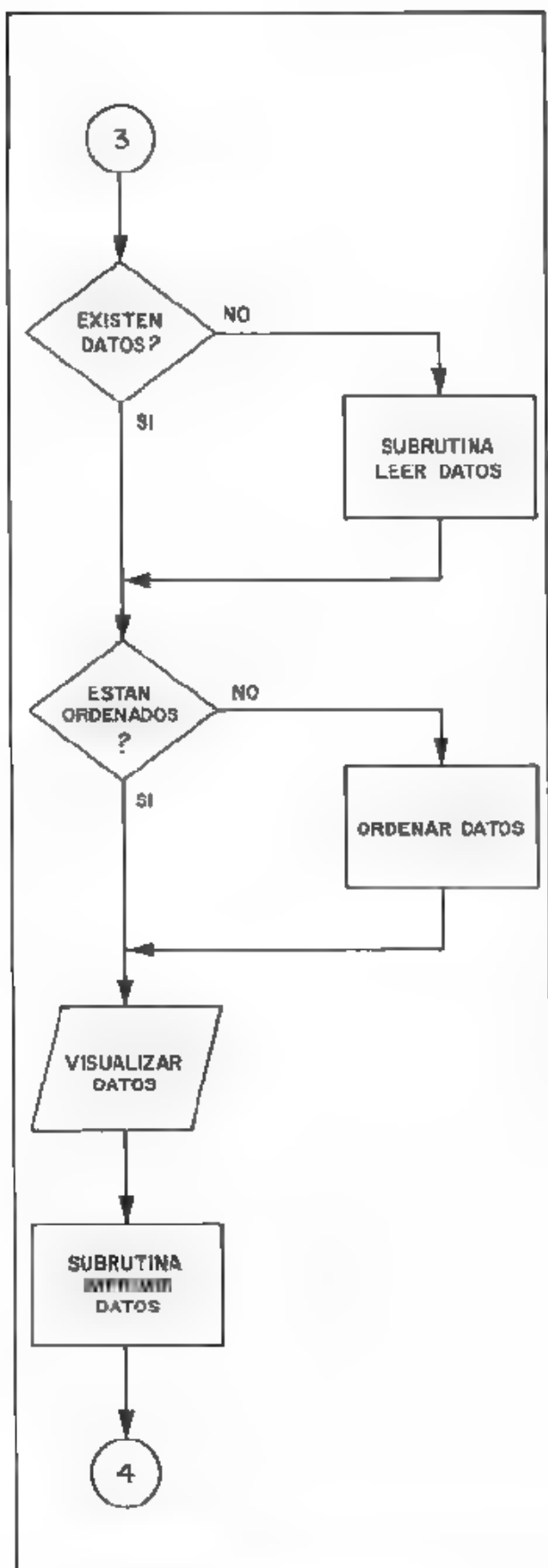
VISUALIZACION DEL «LISTIN»

ENTRADA	PROCESO	SALIDA
INTRODUCCION DE LA OPCION	SELECCION	PRESENTACION DE LA OPCION

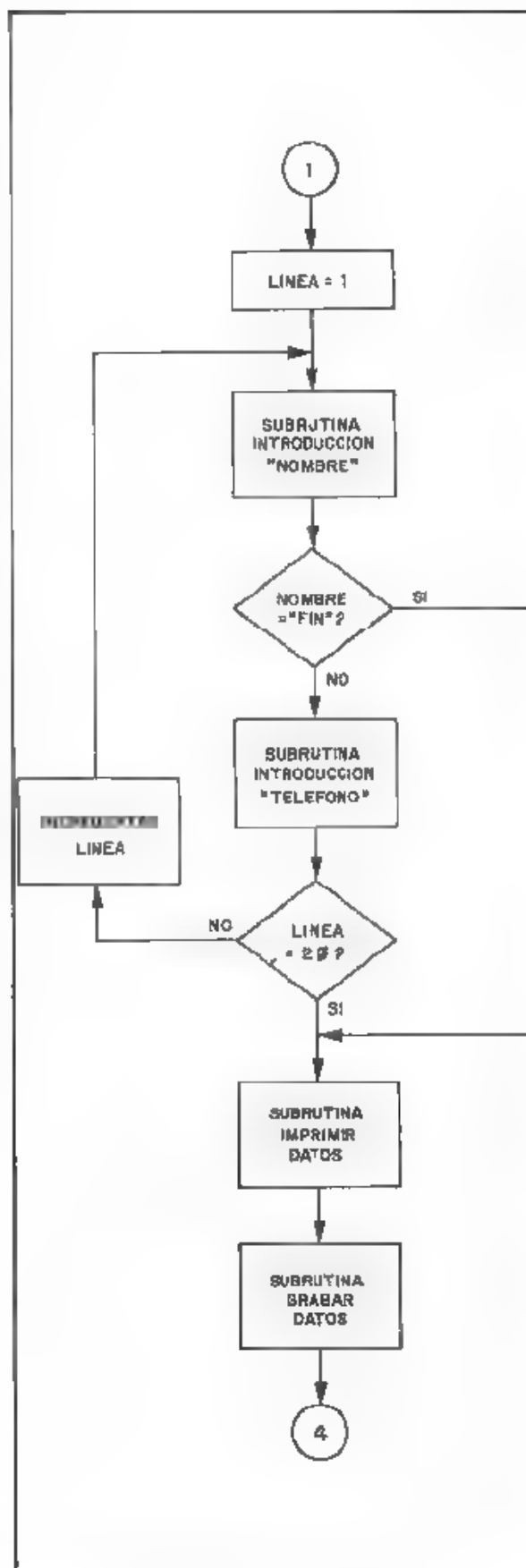
MENU DE OPCIONES



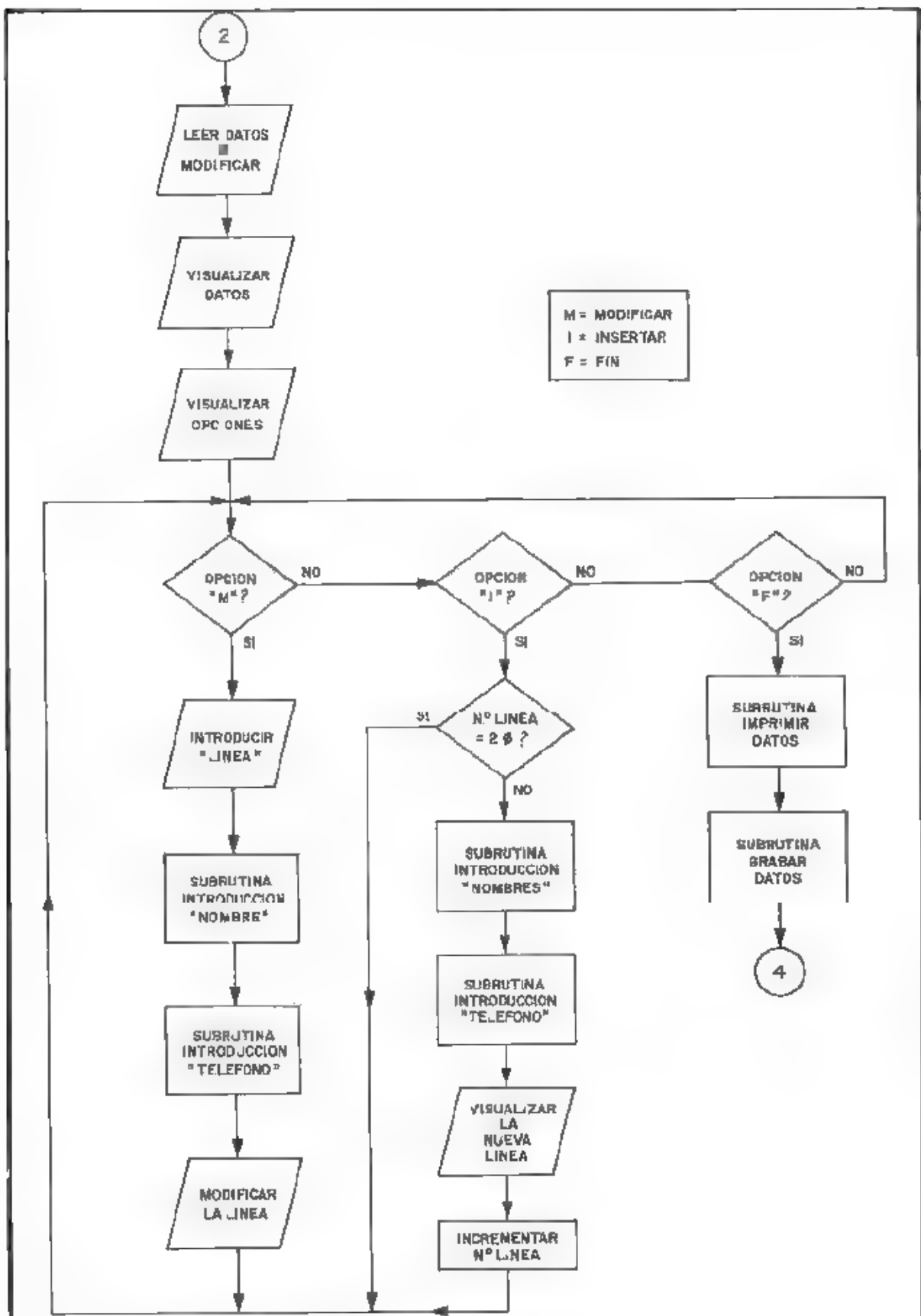
Menú de opciones.



Visualización del «listín telefónico» ordenado.



Editar un nuevo «listín telefónico».



Modificar un «listín telefónico» ya existente.

```

*****
2010 FOR X=1 TO 20
2020 IF N$(X)=""
    THEN GO TO 2040
2030 NEXT X
2040 LET ultimo=x-1
2050 RETURN
5000 REM
*****
*
* LISTADO POR *
*
* IMPRESORA *
*
*****
5005 POKE FLAG$2,8
5010 PRINT #0,AT 0,0,"Desea un l
istado (S/N)"
5020 PAUSE 0. LET P$=INKEY$
5030 IF P$="S" THEN BEEP 0.05,20
GO TO 5060
5040 IF P$="N" THEN BEEP 0.05,20
RETURN
5050 BEEP 0.2,-15 GO TO 5020
5060 PRINT #0,AT 0,0,"Ponga en f
uncionamiento la impresora y pul
se una tecla."
5070 PAUSE 0 BEEP 0.05,20
5080 COPY
5090 PRINT #0,AT 0,0,"Impresion
terminada."
"
5100 FOR x=1 TO 200 NEXT x
5110 GO TO 5010
5200 REM
*****
*
* GRABACION EN *
*
* CINTA *
*
*****
5210 PRINT #0,AT 0,0,"Coloque un

```

```

a cinta en el cassette y grabe lo
s datos."
5220 FOR x=1 TO 400 NEXT x
5225 PRINT #0,AT 0,0 "
"
5230 SAVE "telefono" DATA N$(1)
5240 PRINT #0,AT 0,0 "Grabacion
terminada."
5250 FOR x=1 TO 200 NEXT x
5260 PRINT #0,AT 0,0,"Rebobine l
a cinta (verificacion)"
5270 VERIFY "telefono" DATA N$(1)
5280 CLS
5290 PRINT #0,AT 0,0 "Grabacion
correcta"
5300 FOR x=1 TO 200 NEXT x
5305 POKE FLAG$2,8
5310 PRINT #0,AT 0,0,"Desea grab
ar otra cinta (S/N)"
5320 PAUSE 0. LET P$=INKEY$
5330 IF P$="S" THEN BEEP 0.05,20
GO TO 5200
5340 IF P$="N" THEN BEEP 0.05,20
CLS RETURN
5350 BEEP 0.2,-15. GO TO 5320
5500 REM
*****
*
* LECTURA DE *
*
* DATOS *
*
*****
5505 CLS
5510 PRINT #0,AT 0,0,"Coloque la
cinta con los datos."
5520 LOAD "telefono" DATA N$(1)
5530 CLS
5540 PRINT #0,AT 0,0;"Lectura co
rrecta."
5550 FOR x=1 TO 200 NEXT x
5560 RETURN

```

El juego de sentencias

Después de haber repasado en capítulos anteriores los conocimientos esenciales del lenguaje de programación BASIC, empezamos en éste a dar una visión global del juego de sentencias del Spectrum, posteriormente irán siendo estudiadas una a una.

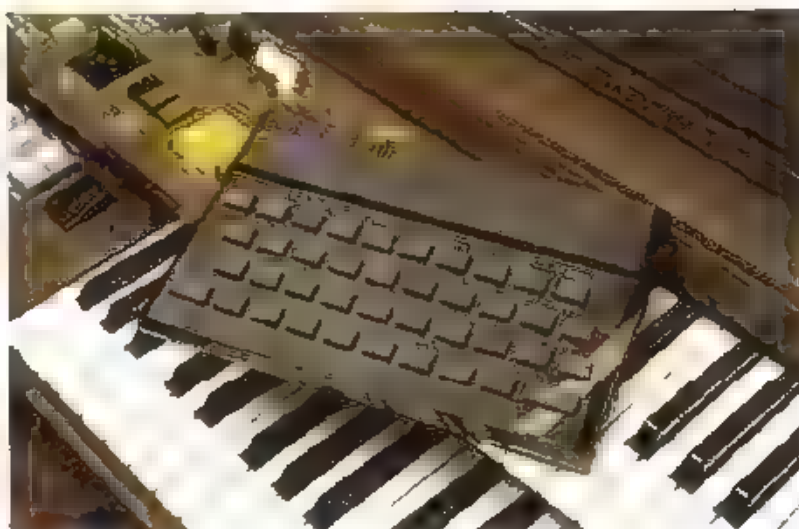
Clasificación

El juego de comandos, sentencias y funciones, consta de 88 palabras claves, éstas pueden clasificarse, según su funcionalidad, de la siguiente forma:

- Comandos de control.
- Comandos de programación.
- Comandos de entrada/salida.
- Manejo de cadenas.
- Funciones aritméticas.
- Funciones lógicas.
- Comandos de dibujo.
- Comandos de control de color e impresión.
- Sonido.
- Manejo impresora.
- Manejo periférico (Interfaze - I).
- Auxiliares.

Comandos de control

Esta serie de comandos normalmente son introducidos de forma directa en el ordenador, es decir, que no forman parte de las instrucciones de un programa, aunque en algunas ocasiones sí pueden formar.



Algunas de las funciones de estos comandos son.

- Ejecución de un programa.
- Listado por pantalla de las instrucciones.
- Grabación de programas.
- Carga de programas.
- Borrado de pantalla.
- Borrado de memoria.
- Etc.

Comandos de programación

La función principal de estas sentencias es, manejar la información interna, realizar cálculos, tomar decisiones, etc. Algunas de las funciones más concretas son.

- Asignación de variables.
- Dimensionado de Matrices.
- Parada en la ejecución de un programa.
- Temporizaciones.
- Control de bucles.

- Decisiones.
- Salto a una línea determinada.
- Salto a subrutina.
- Etc.

Comandos de entrada/salida

Son todos aquellos que indican al ordenador qué tipo de operación de lectura o escritura de datos debe realizar, como por ejemplo.

- Visualización en pantalla.
- Entrada de datos por teclado.
- Reconocimiento de la última tecla pulsada.
- Lectura de los datos de una tabla.
- Escritura en una posición de memoria.
- Lectura de una posición de memoria.
- Entrada o salida de datos por el conector trasero.

Manejo de las cadenas

Útiles en el manejo de cadenas como:

- Conocer la longitud de una cadena.
- Asignar a una variable numérica el valor de una cadena.
- Pasar a decima el valor de un carácter ASCII.
- Pasar a Código ASCII un número decimal.
- Etc...

Funciones aritméticas

Con esta serie de funciones, el Spectrum se convierte en una útil calculadora científica, ya que permite efectuar las siguientes operaciones:

- Raíz cuadrada.
- Logaritmos naturales.
- Seno.
- Coseno.
- Tangente.
- Arco seno.
- Arco coseno.
- Arco tangente.
- Generación de números aleatorios.
- Conversión de binario a decimal.
- Conocer el signo de una expresión.
- Averiguar su valor absoluto o su parte entera.
- Operar con las constantes matemáticas «pi» y «e».

Funciones lógicas

Al estudiar los operadores lógicos, fueron repasadas estas funciones:

- AND.
- OR.
- NOT.

Comandos de dibujo

Con estas sentencias se puede realizar cualquier tipo de gráfico en alta resolución. Se puede dibujar mediante

PROGRAMA 1

```
10 REM *****
   CURSOS
   BASIC/SINCLAIR
   *****
   "COLOREAR"
   *****

20 BORDER 1: PAPER 1: CLS
30 PRINT #0; AT 1,0; "CURSO BASI
C/SINCLAIR MICROHOBBY"
40 FOR y=0 TO 21
50 FOR x=0 TO 31
60 LET grafico=INT (RND*15)+12
9
70 LET color papel=RND*7
80 LET color tinta=RND*7
90 PRINT AT y,x; PAPER color p
apel; INK color tinta; CHR$ grafi
co
100 NEXT x
110 LET color borde=RND*7
120 BORDER color borde
130 NEXT y
140 PRINT AT 10,4; PAPER 9, INK
8; "EL COLOR EN EL SPECTRUM"
150 PAUSE 0
160 BORDER 7: PAPER 7: INK 0: C
LS
```



- Puntos.
 - Líneas.
 - Circunferencias.
 - Arcos de circunferencia.
- Los argumentos usuales que hay que indicar en este tipo de sentencias son las coor-

denadas «X» e «Y» de la posición a dibujar.

También existe la posibilidad de realizar gráficos, utilizando los *predefinidos*, que incorpora el Spectrum, y los *definidos* por el usuario (GDU).

PROGRAMA 2

```

10 REM *****
   *          *
   *    CURSO    *
   * BASIC/SINCLAIR *
   *          *
   * ***** *
   * "GRAFICAS" *
   *          *
   * ***** *

15 BORDER 2: PAPER 2: INK 7: C
LS
20 DRAW 255,0: DRAW 0,175: DRA
W -255,0: DRAW 0,-175
25 PRINT AT 5,7;"PROGRAMA ""GR
AFICAS""
30 PRINT FLASH 1,AT 12,9,"PARE
LA CINTA"
35 PLOT 0,23: DRAW 255,0
40 PRINT AT 20,1," ";
42 RESTORE
45 FOR X=1 TO 26
50 READ logo
55 PRINT CHR$(logo);: BEEP 0.05
,logo/2
60 NEXT x: GO SUB 750
65 DATA 127,32,77,73,67,82,79,
72,79,66,66,89,32,38,32,82,65,70
,65,69,76,32,80,82,65,68,69,83
70 PRINT AT 12,9,"
75 PRINT #0,AT 1,1;"Pulse una
tecla para continuar"
80 PAUSE 0
85 REM *****
   *          *
   * DIBUJO *
   *          *
   * ***** *

90 BORDER 1: PAPER 1: INK 6: C
LS
100 FOR x=-127 TO 128 STEP 2
110 PLOT OVER 1,127,0
120 DRAW OVER 1;x,175
130 NEXT x
140 GO SUB 750
200 FOR x=-127 TO 128 STEP 2
210 PLOT OVER 1;127,175
220 DRAW OVER 1;-x,-175
230 NEXT x
240 GO SUB 750
300 FOR y=88 TO -87 STEP -2
310 PLOT OVER 1;0,87
320 DRAW OVER 1,255,y
330 NEXT y
340 GO SUB 750
400 FOR y=88 TO -87 STEP -2
410 PLOT OVER 1;255,87
420 DRAW OVER 1;-255,y
430 NEXT y
440 GO SUB 750

```

Comandos de control de color

Pueden definirse hasta ocho colores básicos con estos comandos, aunque pueden simularse hasta 56 más. Con ellos se puede alterar el color de:

- El *borde* de la pantalla (zona donde no se imprimen caracteres)
- El *fondo*, también conocido como *papel* (zona en la cual se visualizan los caracteres o los gráficos).
- El color del propio carácter o gráfico, también conocido como *tinta*.

Estos colores pueden ser controlados en dos gamas de brillo.

La impresión se puede variar con otra serie de comandos que controlan.

- El parpadeo de los caracteres.
- La inversión del color de *tinta* por el de *papel*.
- La sobreimpresión de caracteres.

Sonido

El Spectrum tiene, con el empleo de una sola instrucción, capacidad para producir una amplia variedad de notas musicales; esta posibilidad es bastante interesante, ya que nos permite, con su uso, amenizar los programas. Estos sonidos son escuchados a través del altavoz interno, aunque también es posible enchufar un amplificador a los conectores EAR o MIC para incrementar su volumen.

Manejo impresora

Puede incorporarse una impresora ZX u otra, con el *interface* adecuado, al conector de

```

500 FOR y=175 TO 0 STEP -2
510 PLOT OVER 1;0,y
520 DRAW OVER 1;255,0
530 NEXT y
540 GO SUB 750
600 FOR x=0 TO 255 STEP 1
610 PLOT OVER 1;x,0
620 DRAW OVER 1;0,175
630 NEXT x
640 GO SUB 750
700 GO TO 100
740 REM

```

```

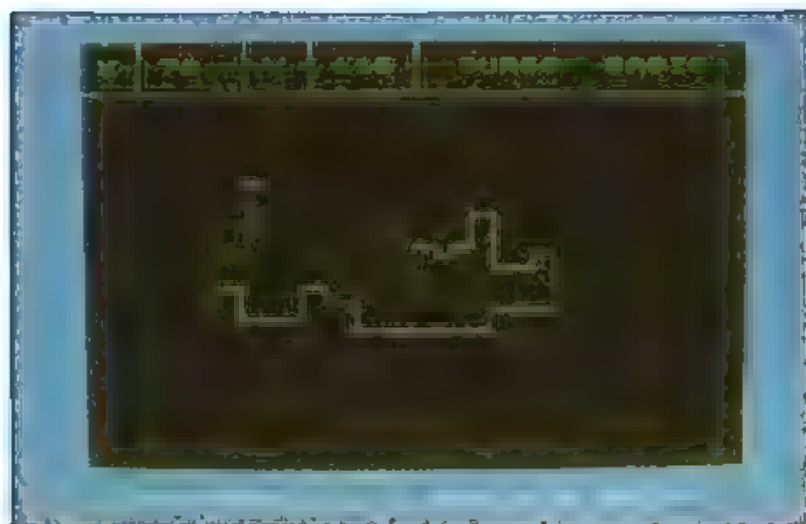
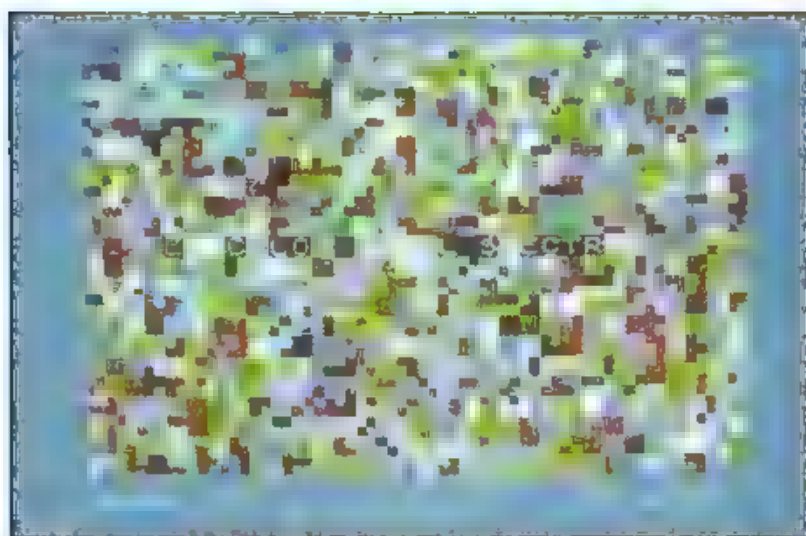
*****
*          *
*  RETARDO  *
*          *
*****

```

```

750 FOR z=0 TO 300
760 NEXT z
770 RETURN

```



expansión trasera. En ésta se pueden obtener:

- Listado de programas.
- Copia de los gráficos presentes en pantalla.
- Mensajes.

INTERFACE - 1

El INTERFACE 1 es un dispositivo que también se adapta al conector de expansión del Spectrum. Con el manejo de ciertas instrucciones y a través de este interface se puede controlar:

- Un máximo de ocho *Microdrive* (Unidades de almacenamiento).
- La *red local* (mediante la cual se pueden atender hasta 64 Spectrum).
- Salida *RS-232* (para conectar periféricos que utilicen este sistema de transmisión de datos en serie).

También prolonga el conector de expansión de manera que se puedan añadir más periféricos, aunque esté conectado el INTERFACE - 1.

Manejo Microdrive

A través del INTERFACE - 1 se pueden realizar las siguientes operaciones con los Microdrive:

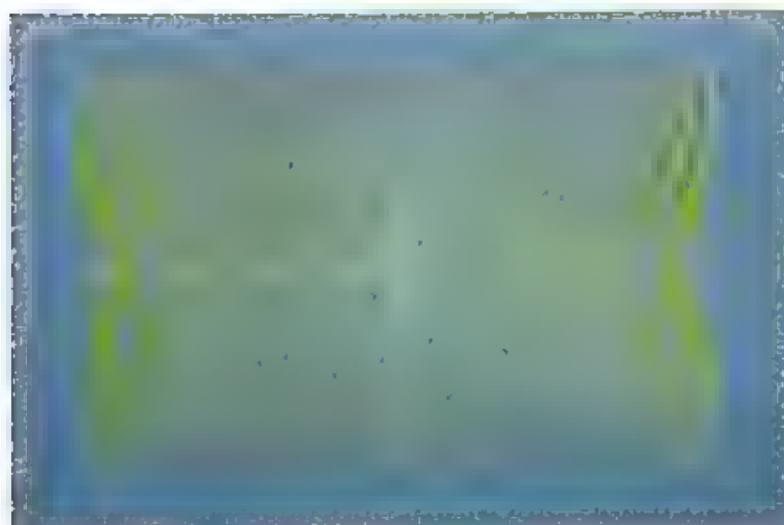
- Formatear o inicializar cartuchos.
- Almacenar y verificar programas en cartucho.
- Cargar programas.
- Borrar ficheros de datos y programas.
- Combinar programas residentes en memoria con los archivados en cartucho.
- Catalogar cartuchos, es decir, obtener un directorio o listado de los programas contenidos en él.
- Abrir y cerrar ficheros de datos.

PROGRAMA 3

```

10 REM *****
*          CURSO          *
* BASIC/SINCLAIR         *
* *****               *
*          G.D.U.         *
* *****               *
20 BORDER 4: PAPER 4: INK 0: C
LS
30 PRINT AT 10,7;"Espere un mo
mento"
40 REM *****
*          *
* GENERACION *
* *****
50 FOR n=USR "A" TO LSR "U"+7
60 LET GDU=INT (RND*256)
70 POKE n,GDU
80 NEXT n
90 REM *****
*          *
* VISUALIZACION *
*          *
* *****
100 CLS
110 FOR n=0 TO 9
120 PRINT CHR$ (144+n);"
";CHR$ (65+n),CHR$ (154+n);"
";CHR$ (75+n)
130 NEXT n
140 PRINT AT 21,8;CHR$ 164;"
";CHR$ 85
150 PAUSE 0

```



- Grabar y leer datos de un fichero.
- Mover datos entre ficheros.

Auxiliares

Con estas sentencias se puede controlar el formato de

impresión de alguno de los comandos de entrada/salida; también, por ejemplo, se puede conocer si un punto determinado de la pantalla tiene color de *tinta* o *papel*.

Programas

Los párrafos siguientes son una breve descripción de los programas que acompañan este capítulo, y que vienen a mostrar la capacidad del Spectrum para generar colores, gráficos y sonido.

Programa «COLOREAR»

Este programa muestra en pantalla la capacidad del Spectrum para producir colores en *baja resolución*. Los gráficos predefinidos son generados aleatoriamente, al igual que los colores de borde, papel y tinta.

Una vez ejecutado, pulse cualquier tecla para terminar y aparecerá el mensaje:

OK,160 4

Cada vez que se ejecuta se obtienen resultados distintos.

Programa «GRAFICAS»

Visualiza en pantalla unas gráficas realizadas en *alta resolución*, empleando la técnica de sobreimpresión. Una vez representadas, el programa las vuelve a generar; para salir de este bucle, apriete simultáneamente las teclas «CAPS SHIFT» y «SPACE», el ordenador nos presentará el mensaje:

I BREAK into program

El programa debe ser grabado de la forma:

SAVE "GRAFICAS" LINE 10

PROGRAMA 4

```

10 REM *****
  *          *
  *      CURSO      *
  *          *
  * BASIC/SINCLAIR  *
  *          *
  * *****        *
  *          *
  *      DISUJANDO  *
  *          *
  * *****        *
20 BORDER 1 PAPER 0 INK 7 C
L3
30 RESTORE 80
40 FOR X=1 TO 20
50 READ caracter
60 PRINT AT 20,(1+x),CHR$(carac
car, BEEP 0.05,caracter/2
70 NEXT X
80 DATA 127,32,77,73,67,82,79,
72,79,66,66,89,32,38,32,82,65,70,
65,69,76,32,80,82,65,68,69,63
110 REM *****
  *          *
  * PIZARRA      *
  *          *
  * *****
120 DRAW 0,175 DRAW 255,0 DRA
W 0,-175 DRAW -255,0
130 REM *****
  *          *
  *      TIZA      *
  *          *
  * *****
132 PRINT AT 3,11,"DISUJANDO"
134 PRINT AT 4,11,""
140 PRINT AT 8,10; INK 4,"U. - U
ERDE"
150 PRINT AT 10,10; INK 5,"C.
CYAN"
160 PRINT AT 12,10; INK 6;"A.-
AMARILLO"
170 PRINT AT 14,10; INK 7;"B.-
BLANCO"
180 PRINT #0,AT 1,0,"Color de l
a tiza"
185 POKE 23558.8
190 PAUSE 0
200 LET A$=INKEY$
210 IF A$="U" THEN LET color=4
GO TO 260
220 IF A$="C" THEN LET color=5
GO TO 260
230 IF A$="A" THEN LET color=6-
GO TO 260
240 IF A$="B" THEN LET color=7
GO TO 260
250 BEEP 0.2,-15. GO TO 190
260 REM *****
  *          *
  * POSICION      *
  *          *
  * INICIAL        *
  *          *
  * *****
270 BEEP 0.2,20
280 INK color: CLS
290 DRAW 0,175 DRAW 255 0 DRA
W 0,-175 DRAW -255,0

```

```

300 PLOT 0,152 DRAW 255,0
310 PLOT 24,152 DRAW 0,22
320 PLOT 54,152 DRAW 0,22
330 PLOT 88,152 DRAW 0,22
340 PLOT 128,152 DRAW 0,22
350 PRINT AT 1,1,"X" AT 1,9,"Y"
360 PRINT AT 1,18,"CURSO BASIC"
370 LET pos x=127
380 LET pos y=80
390 PRINT AT 1,4,pos x,AT 1,12,
pos y
392 PRINT #0,"CURSOSES=dibuja/
B-borra,F-funcion"
400 REM *****
  *          *
  * DIBUJAR CON    *
  *          *
  * CURSOSES        *
  *          *
  * *****
410 PLOT pos x,pos y
420 IF INKEY$="5" THEN LET pos
x=pos x-1: LET movimiento=1 GO
TO 490
430 IF INKEY$="6" THEN LET pos
y=pos y-1: LET movimiento=0 GO
TO 490
440 IF INKEY$="7" THEN LET pos
y=pos y+1: LET movimiento=0 GO
TO 490
450 IF INKEY$="8" THEN LET pos
x=pos x+1: LET movimiento=1. GO
TO 490
460 IF INKEY$="B" THEN BEEP 0.0
5,20 CLS GO TO 260
470 IF INKEY$="F" THEN BEEP 0.5
,GO STOP
480 GO TO 420
490 GO SUB 700
500 IF borde=1 THEN GO TO 420
510 GO SUB 1000 GO TO 410
700 REM *****
  *          *
  * VERIFICACION   *
  *          *
  * *****
710 IF pos x<1 THEN LET pos x=p
os x+1 BEEP 0.05,20 GO TO 420
720 IF pos x>254 THEN LET pos x
=pos x-1 BEEP 0.05,20 GO TO 70
0
730 IF pos y<1 THEN LET pos y=p
os y+1 BEEP 0.05,20 GO TO 700
740 IF pos y>151 THEN LET pos y
=pos y-1. BEEP 0.05,20 GO TO 70
0
750 LET borde=0 RETURN
760 LET borde=1 RETURN
1000 REM *****
  *          *
  * VISUALIZACION  *
  *          *
  * *****
1010 IF movimiento=1 THEN PRINT
AT 1,4 PRINT AT 1,4 pos x
RETURN
1020 PRINT AT 1,12," " PRINT
AT 1,12,pos y RET RN
1030 RETURN

```

Programa "GDU"

Con este programa se generan 21 gráficos definidos por usuario (GDU), de una forma aleatoria. Al lado de cada gráfico aparece la letra a la que queda asignado.

Una vez ejecutado pulse cualquier tecla para terminar y aparecerá el mensaje:

Pase a modo **G** y pulsando una tecla de la «A» a la «U», aparecerá en pantalla el gráfico correspondiente.

Programa "DIBUANDO"

Este programa permite realizar cualquier tipo de dibujo

en la pantalla, utilizando los cursores de movimiento situados en las teclas «5», «6», «7» y «8». Al ser ejecutado, lo primero que debemos hacer es pulsar una de las cuatro teclas que indican el color de la «tiza» (V = verde, C = cyan, A = amarillo, B = blanco).

En la parte superior izquierda de la pantalla aparece en

PROGRAMA 5

```

10 REM *****
*
*      CURSO
*
* BASIC/SINCLAIR
*
*****
*
*      "BIPBIP"
*
*****

15 BORDER 2 PAPER 2 INK 6 0
LS
20 DRAW 255 0 DRAW 0,175: DRA
W 255 0 DRAW 0,175
25 PRINT AT 5,7: "PROGRAMA ""BI
PBIP""
30 PRINT FLASH 1, AT 12,9, "PARE
LA CINTA"
35 PLOT 0,23 DRAW 255,0
40 PRINT AT 20,1, " "
42 RESTORE
45 FOR X=1 TO 20
50 READ LOGO
55 PRINT CHR$(LOGO): RFPF @ 05
60 NEXT X: GO SUB 90
65 DATA 127,32,77,73,67,82,79,
72,79,66,60,89,32,30,32,82,65,70,
65,69,76,32,80,82,65,68,69,83
70 PRINT AT 12,9, " "
72 GO TO 100
84 REM
*****
*
*      TEMPORIZACION
*
*****

90 FOR X 1 TO 300 NEXT X: RET
URN
100 GO SUB 120: GO TO 400

120 REM
*****
*
*      TECLADO
*
*****

140 FOR Y=12 TO 14
150 FOR X=12 TO 18
160 PRINT AT Y,X, PAPER 7: " "
170 NEXT X
180 NEXT Y
185 INK 0
190 PLOT 96,79
200 DRAW 56 0 DRAW 0,-23 DRAW
56 0 DRAW 0,23
210 FOR X=103 TO 143 STEP 8
220 PLOT X 79 DRAW 0, 22
230 NEXT X
240 IF Y=79
250 LET X=100: GO SUB 350
260 LET X=106 GO SUB 350
270 LET X=124 GO SUB 350
280 LET X=132 GO SUB 350
290 LET X=140 GO SUB 350
340 RETURN
350 FOR X=X TO X+6
360 PLOT X Y DRAW 0,-15
370 NEXT X
380 RETURN
400 REM
*****
*
*      MUSICA
*
*****

430 LET clave de sol=12
440 LET duracion=0.2
450 FOR X=1 TO 22
460 READ nota, pausa
480 IF nota=0 THEN LET posicion
=12
490 IF nota=2 THEN LET posicion
=13
500 IF nota=4 THEN LET posicion
=14
510 IF nota=5 THEN LET posicion
=15
520 IF nota=7 THEN LET posicion
=16
530 PRINT AT 15, posicion " "
532 BEEP duracion, nota+clave de
sol
534 PRINT AT 15,12, " "
540 PAUSE pausa
550 NEXT X
560 DATA 4,2,4,2,4,10,4,2,4,2,4,
10,4,2,7,2,0,2,2,4,10,0,2,5,2,
5,10,4,2,4,2,4,10,7,2,5,2,4,2,2,
2,0,10
600 PRINT NO, AT 1,1, "Pulse una
tecla para continuar"
610 PAUSE 0
615 REM
*****
*
*      TECLADO MUSICAL
*
*****

620 BORDER 1 PAPER 1 INK 6 0
LS
630 PRINT A partir de este
momento su Spectrum se ha conve
rtido en un instrumento musical.
Obtendra una nota
diferente cada vez que pulse u
na tecla. Simultaneamente pued
e utilizar la tecla CAPS SHIF
T o SYMBOL SHIFT
640 PRINT NO, Pulse CAPS SHIF
T + SPACE para terminar.
650 GO SUB 120
670 LET A$=IN $
680 IF A$= ' THEN GO TO 670
700 LET nota=CODE A$,4
710 BEEP duracion nota
720 GO TO 670

```

CLASIFICACION DE LAS SENTENCIAS SEGUN SU FUNCION

COMANDOS DE CONTROL

RUN
CONT
LIST
LOAD
SAVE
VERIFY
MERGE
CLEAR
CLS
NEW

MANEJO DE CADENAS

VAL
VALS
LEN
SIHS
CHRS
CODE

COMANDOS DE CONTROL DE COLOR E IMPRESION

BORDER
PAPER
NK
BRIGHT
INVERSE
FLASH
OVER

COMANDOS DE PROGRAMACION

REM
LET
DIM
DEF FN
PAUSE
STOP
FOR-TO-STEP
NEXT
IF-THEN
GO TO
GO SUB
RETURN
RESTORE
RAND
USR

FUNCIONES ARITMETICAS

SGN
ABS
INT
LN
EXP
SQR
FN
RND
SIN
COS
TAN
ASN
ACS
ATN
PI
BIN

AUXILIARES

LINE
TAB
AT
POINT
ATTR
SCREENS

SONIDO

BEEP

MANEJO IMPRESORA

LLST
LPRINT
COPY

COMANDOS DE ENTRADA/SALIDA

PRINT
INPUT
INKEY\$
READ
DATA
PEEK
POKE
IN
OUT

FUNCIONES LOGICAS

AND
OR
NOT

COMANDOS DE DIBUJO

PLOT
DRAW
CIRCLE

MANEJO PERIFERICOS (INTERFACE-1)

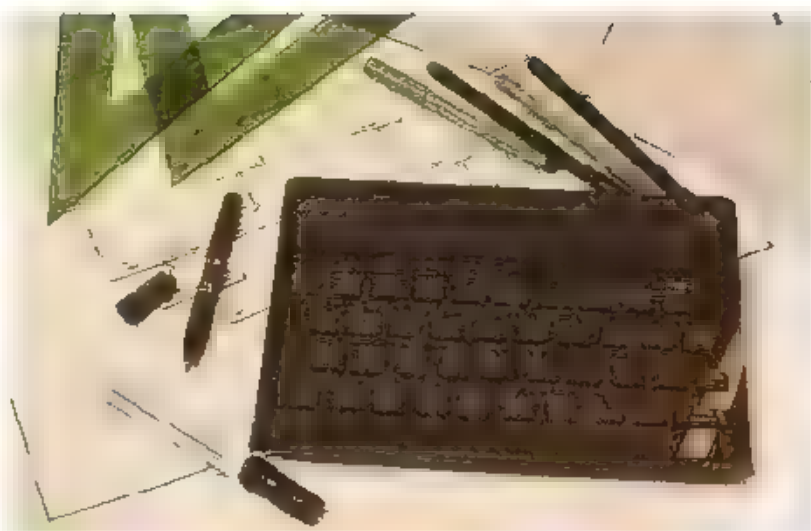
FORMAT
CAT
ERASE
OPEN
CLOSE
MOVE

tiempo real las coordenadas de la «tiza». La opción «B» borra toda la pantalla y posiciona la «tiza» en las coordenadas iniciales $X = 127$ e $Y = 80$. La opción «F» permite salir del programa, apareciendo el mensaje.

9 STOP statement. 470 3

Para que se autoejecute, grábalo de la forma.

SAVE "DIBUJANDO" LINE 10



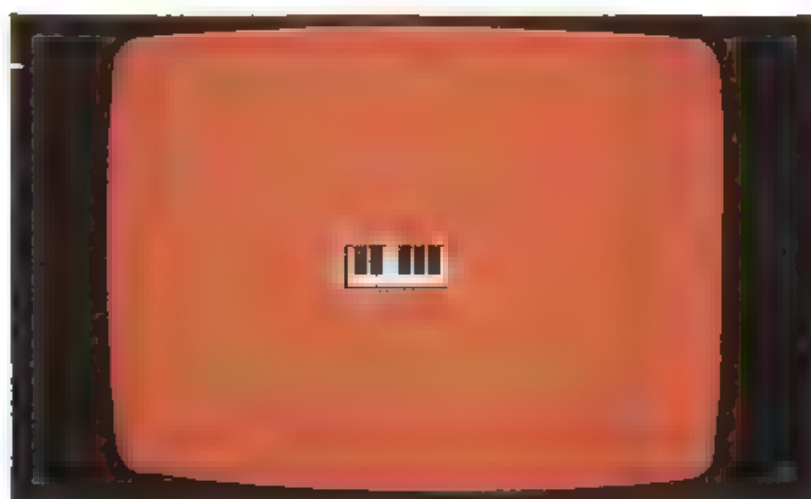
Programa "BIPBIP"

El último programa presentado convierte a su Spectrum en un instrumento musical.

Después de la presentación se genera una melodía conocida por todos. Las instrucciones de manejo se encuentran en el propio programa.

Sávelo de la siguiente manera:

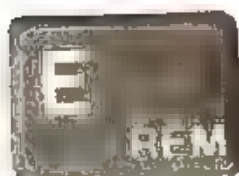
SAVE "BIPBIP" LINE 10



COMANDOS BASICOS

REM

Acceso al teclado



MODO K

Tipo de sentencia

Comando de programación

Concepto

Si durante la ejecución de un programa el ordenador detecta una sentencia REM, automáticamente analiza la siguiente instrucción y no ejecuta la correspondiente a REM. ¿Para qué sirve entonces una sentencia que no se ejecuta?, simplemente para poder introducir líneas de comentario (REMARK en inglés) dentro de un programa.

Estos comentarios sirven, por ejemplo, para indicar al principio de un programa el título de éste, el nombre del programador, la fecha y la edición, este último dato es bastante importante ya que nos permite comparar a simple vista, cuál es la versión actualizada de un mismo programa.

Dentro del programa, los comentarios sirven para indicarnos las funciones que realizan las distintas rutinas de que se compone o para aclarar el sig-

nificado de alguna de las variables utilizadas.

La estructura de esta sentencia es la siguiente:

SENTENCIA	ARGUMENTO
REM	Cualquier carácter

Ejemplos:

— Título de programas.

```
10 REM ***** MI MICROHOBBY *****
20 REM
30 REM      @ Pab o Otero
40 REM      Agosto 84
50 REM      Edición = 5
60 REM *****
```

Función de una rutina.

200 REM ** CALCULO VARIABLES **

— Significado de variables.

```
200 REM 22+44577777777777777777
201 REM
202 REM  VALX=COORDENADA X
203 REM
204 REM  VALY=COORDENADA Y
205 REM
206 REM  N$ = Nombre
207 REM
208 REM *****
```

Las sentencias REM ayudan a que un programa tenga claridad y limpieza, ya que si al cabo de un tiempo debemos realizar una modificación, será más fácil realizarla en un programa que esté documentado con comentarios, que en otro que no lo esté.

Consideraciones

En la edición de sentencias REM es necesario tener en cuenta los siguientes puntos:

— Para localizar con facilidad las distintas rutinas de un programa conviene que es-

tas sentencias resalten sobre las demás. Se puede, por ejemplo, enmarcar los mensajes con asteriscos (*) u otro símbolo, también puede utilizarse la función de vídeo invertido (INV. VIDEO), ésta será explicada posteriormente.

— No es necesario encerrar el argumento entre comillas (" "), aunque esté formado por una cadena alfanumérica.

Pueden ir en líneas independientes o al final de una cadena de sentencias.

Ejemplo:

```
200 PRINT " - MICROHOBBY " REM
400 Esto es un ejemplo de REM
```

ADVERTENCIA

A continuación de este tipo de sentencias, no debe editarse otra instrucción en la misma línea. Como el símbolo utilizado como separador de instrucciones (:) también puede formar parte del argumento de una sentencia REM, la sentencia editada a continuación no sería ejecutada.

Ejemplo:

```
200 REM " - MICROHOBBY " REM
400 PRINT " - MICROHOBBY " REM
```

Si ejecutamos esta instrucción, resulta que la cadena «—MICROHOBBY » no será visualizada, ya que el ordenador interpreta que forma par-

te de la sentencia REM y por lo tanto no se ejecuta.

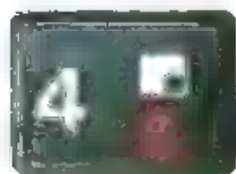
- Un argumento largo puede ser incluido en una sola línea.

Ejemplo,



Las sentencias REM ocupan parte de la memoria del ordenador, aunque como no son procesadas tienen la ventaja de no retardar la ejecución de un programa.

Funciones de Video



La función que el Spectrum presenta por defecto es la de *video normal* (TRUE VIDEO), es decir, que los caracteres se visualizan en el color de la tinta (INK) y el fondo en el del papel (PAPER). Si desea que estos dos colores se intercambien para destacar un fragmento de programa o algún texto, como, por ejemplo, el argumento de una sentencia REM, es necesario utilizar la función de *video invertido* (INV VIDEO).

A esta función se accede,

en la edición de instrucciones, después de haber introducido el número de línea, pulsando la tecla CAPS SHIFT simultáneamente con la tecla correspondiente al número 4. A partir de este momento los caracteres tendrán el color del papel y el fondo el color de la tinta.

Para retornar a la visualización en video normal es necesario pulsar CAPS SHIFT y la tecla número 3, simultáneamente.

ADVERTENCIA

Para que este retorno tenga validez es necesario realizarlo dentro de una instrucción, es decir, después de haber introducido el número de línea y antes de pulsar ENTER, bien en la instrucción en la que se insertó la función invertida de video, bien en otra posterior.

Un ejemplo del manejo de estas funciones es el siguiente:



LET

Acceso al teclado



MODO K

Tipo de sentencia

Comando de programación

Definición

También es conocido como comando de *asignación* ya que a una variable (numérica o de cadena) le asigna un valor; este puede ser una constante o variable (numérica o de cadena), el resultado de una expresión matemática, una operación de cadena o una función VAL, esta última será vista en otro capítulo.

Deben tenerse en cuenta las siguientes consideraciones:

- Una variable o constante numérica y una expresión matemática, sólo pueden ser asignadas a una variable numérica.
- Una variable o constante alfanumérica, una operación de cadena (concatenación, fragmentación) o una función VAL pueden asignarse solamente a las funciones de cadena.

Las estructuras de esta sentencia son

a)	SENTENCIA	ARGUMENTO
	LET	Variable - expresión

Ejemplos:

```

10 LET dato=123
20 LET dni=215187
30 LET resultado=0.0
40 LET resultado=7
  
```

b)	SENTENCIA	ARGUMENTO
	LET	Variable \$ - expresión de cadena

Ejemplos:

```

10 LET $= MICROHOBBY
20 LET $= SEMANA
30 LET $= $405
40 LET $= TO $ BASIC
50 LET $412=0
  
```

Para utilizar una variable dentro de una expresión, debe estar asignada previamente por una sentencia LET, es decir, si las variables «dato» y «a» no hubieran sido asignadas no se podría haber utilizado la sentencia:

```
30 LET resultado = dato + a
```

ya que hubiera dado el siguiente error

```
2 Variable not found
```

lo mismo ocurre con la sentencia

```
30 LET c$ = a$ + b$
```

Veamos, por pasos, otro ejemplo de utilización de la sentencia LET. Supongamos que se desea implementar la fórmula que calcula el área de un círculo conociendo su diámetro ($d = 10$).

$$S = \pi r^2 \text{ donde } r = \frac{d}{2}$$

1º Asignar a la variable «diámetro» el valor 10.

```
LET diametro = 10
```

2º Calcular el radio ($r = d/2$)

```
LET radio = diametro/2
```

3º Asignación del valor «pi» (π). Podríamos hacer:

```
LET pi = 3.141592
```

pero como el Spectrum dispone de esta constante numérica, no es necesario asignarla. Pi se encuentra en la tecla «M».

4º Cálculo del área.

```
LET superficie = pi * radio^2
```

5º El programa completo sería.

```
10 LET diametro = 10
20 LET radio = diametro/2
30 LET superficie = pi * radio^2
40 PRINT superficie
```

Después de ejecutar este programa la variable «superficie» tendrá almacenado el valor del área de un círculo de diámetro 10; para visualizar este valor incluya la siguiente instrucción y ejecute de nuevo.

```
40 PRINT superficie
```

Modificando el valor de la línea 10, obtendrá diversos valores para la variable «superficie».

Representación gráfica

La visualización de caracteres no ocupa la totalidad de la pantalla de televisor o monitor, ésta sólo se realiza en la zona central, dejando un espacio o borde alrededor (BORDER). La zona de representación permite visualizar 24 líneas de 32 caracteres, las líneas o filas están numeradas de la 0 a la 23 y las columnas de la 0 a la 31 esta disposición permite representar 768 caracteres y se denomina de *baja resolución*, ya que no permite hacer gráficos con calidad aceptable de definición.

Las líneas 0 a 21 son las utilizadas por el usuario y la 22 y 23 las utiliza el ordenador para enviarnos los mensajes y para la introducción de comandos directos y edición de programas; estas líneas también pueden ser utilizadas por el usuario accediendo de un modo especial.

Cada carácter de la zona de usuario puede dividirse en una matriz de 8 por 8 puntos, los cuales se denominan *pixel* (picture element o elementos de imagen), con esta nueva división se obtiene una retícula de 176 por 256 puntos, lo que nos da un total de 45.056 pixel,

esta modalidad se denomina *alta resolución* y permite hacer gráficos con una calidad aceptable de definición.

```
PRINT
```

Acceso al teclado



Tipo de sentencia

Comando de salida.

Definición

Este comando permite visualizar en la pantalla el valor de las constantes, variables, expresiones o textos indicados en el argumento.

Veamos los distintos tipos de estructura que adopta esta sentencia.

a) Visualizar una constante numérica.

SENTENCIA	ARGUMENTO
PRINT	constante

Ejemplos:

```
10 PRINT 50
20 PRINT 3.141592
30 PRINT 10
40 PRINT "PI"
```

b) Visualizar una constante alfanumérica.

SENTENCIA	ARGUMENTO
PRINT	"Cadena"


```

10 PRINT Programa SENIOR
20 PRINT ARCHIVO DE DATOS
30 PRINT Menu de opciones
40 PRINT "Pulse una tecla para"
50 INPUT "continuar"

```

- c) Visualizar variables numéricas previamente asignadas.

SENTENCIA	ARGUMENTO
PRINT	Variable

Ejemplos:

```

10 LET NUMBER=2013
20 PRINT "NAME:"
30 LET ADDRESS="27 RND 1"
40 PRINT "ADDRESS:"
50 LET VAL=1
60 LET TOTAL=VAL*OF=5:10
70 PRINT "TOTAL:"

```

- d) Visualizar variables de cadena asignadas previamente.

SENTENCIA	ARGUMENTO
PRINT	variable \$

Ejemplos

```

10 LET A = 123.45 678
20 PRINT A
30 LET I = 1010
40 PRINT I
50 LET B = "A B C D E F G H I J K L M N O P Q R S T U V W X Y Z"
60 PRINT B

```

- e) Visualizar expresiones aritméticas.

SENTENCIA	ARGUMENTO
PRINT	expresión

Ejemplos:

```
10 LET a=20
20 LET b=7.4
30 PRINT a/b
```

Con esta sentencia y utilizándola como comando directo, es decir, sin atribuirle un número de línea se puede manejar el Spectrum como si fuera una calculadora.

Ejemplos:

```
PRINT 7+4 45
DO J=1,5
PRINT 1, 50 10+31+2+51 2
```

- f) Visualizar operaciones con cadenas.

SENTENCIA	ARGUMENTO
PRINT	expresión \$

Ejemplos:

```

10 LET A$ = "RICHARD L. LENO"
20 IF A$ = "RICHARD L." THEN
30 PRINT A$ + " + b$
40 PRINT A$ + "O b$
50 PRINT A$(5)

```

Desplazamientos

Quando se ejecuta una sentencia del tipo PRINT, el argumento se visualiza al principio de cada línea, en el momento que se completan las 22 líneas y antes de visualizar un nuevo valor, el ordenador nos presenta en la parte inferior de la pantalla, el mensaje:

SCFD 12

con esta pregunta el ordenador queda a la espera de realizar un «scroll» o desplazamiento del texto hacia la parte superior o no. Pulsando cualquiera de las teclas «N», «BREAK» (SPACE) o «STOP» (A) la ejecución del programa se detiene y nos presenta el mensaje:

D BREAK - CONT repeats

si se pulsa cualquier otra tecla, el desplazamiento o *scrolling* se realiza.

El siguiente miniprograma visualiza cien veces, utilizando esta particularidad, a cadena «M CROHOBBY».

40 FOR N=1 TO 100
50 DEF FN F(X)=X
60 NEXT N

Formatos

Por razones de estética o para una presentación de datos más ordenada, se pueden utilizar diversos formatos de visualización.

- a) Visualizar una línea en blanco.

SENTENCIA
PRINT

Ejemplo

```

10 PRINT "Programas"
20 PRINT "PRINT PRINT PRINT"
30 PRINT "Educational"

```

La instrucción 2Ø deja tres espacios en blanco entre el mensaje de la línea 1Ø y el de la 3Ø

- b) Para visualizar variables o constantes seguidas dentro de una misma línea se utiliza el signo ortográfico del punto y coma (;).

Ejemplos.

```
10 LET SEP 11 5:00 PM
20 LET D1813745
30 PRINT "SEP 7 5 51 PM"
```

La instrucción 30 visualiza en pantalla el mensaje:

La suma de $7 + 5$ es 12

[illegible]

- c) Utilizando el signo ortográfico de la coma (,) se consigue una tabulación, de forma automática, de los elementos del argumento.

Con este formato, la pantalla queda dividida, verticalmente en dos zonas o campos, los elementos se van visualizando alternativamente en las columnas 0 y 16.

Cuando la longitud del elemento a visualizar en el primer campo es superior a 15 caracteres, el siguiente se visualiza al comienzo de la línea inferior.

Ejemplo:

```
10 LET A$= "FRUTAS"
20 LET B$= "PIS"
30 PRINT "A B"
40 PRINT "A B"
50 PRINT "Manzanas 50"
60 PRINT "Peras 50"
70 PRINT "Plátanos 11.5"
80 PRINT "Naranjas 1.20"
90 PRINT "Uvas 1.50"
100 PRINT
```

- d) En lugar de utilizar la sentencia PRINT, sin argumento, para dejar líneas en blanco, también puede utilizarse el signo ortográfico del apóstrofe ('), situado en la tecla con el número 7.

El número de espacios en blanco es igual al número de apóstrofes menos uno, ya que el primero sirve para retornar a la línea siguiente.

Ejemplo:

```
10 PRINT "Manzanas 'Peras' "
20 PRINT "Plátanos 'Uvas"
```

TAB

Acceso al teclado

TAB



MODO E

©

Tipo de sentencia

Auxiliar.

Concepto

Esta *palabra clave* se utiliza conjuntamente con PRINT y tabula la salida de datos al valor deseado. Su estructura es la siguiente:

PRINT TAB expresión, valor

una vez evaluada la *expresión*, nos indica ésta el número de columna a partir del cual debe visualizarse el *valor* (numérico o de cadena).

Ejemplos:

```
10 PRINT "AB 10, Dirección"
20 PRINT "CD 123"
30 LET A=10 LET B=10000
40 PRINT TAB 30 A; B
50 LET C= "FASINO"
60 PRINT TAB 20 C
```

Cuando el resultado de la expresión es un número negativo, el ordenador presenta el mensaje:

B integer out of range

Los números decimales son redondeados de manera que TAB 7.5 es igual que TAB 8 y TAB 7.4 es equivalente a TAB 7.

Podríamos suponer en un principio que los posibles valores de la expresión tendrían que estar comprendidos entre 0 y 31, ya que éstos son los números de columna existentes, pues bien, esto no es así, ya que podemos introducir cualquier número comprendido entre 0 y 65535. ¿Cómo tabula el ordenador una sentencia del tipo TAB 750? la respuesta es: reduciendo a «módulo 32», es decir, divide la expresión entre 32 sin obtener decimales y el resto de la división lo interpreta como número de columna.

Observe cómo las siguientes instrucciones, que simulan la forma en que el ordenador calcula el número de columna, realizan la misma función que

PRINT TAB 750, «MICROHOBBY»

0

PRINT TAB 14, «MICROHOBBY»

```
10 REM *** Mod 32 ***
20 PE1
30 LET NUMBER=750
40 LET COLUMN=INT (NUMBER/32)
50 LET REST=NUMBER-COLUMN*32
60 PRINT TAB REST, MICROHOBBY
```

esta sentencia auxiliar puede combinarse con los signos ortográficos (", " y "; ") que también determinan el formato de salida de datos.

Ejemplos:

```
10 PRINT TAB 2, "a"
20 PRINT TAB 8, "b"
30 PRINT TAB 14, "c"
40 PRINT TAB 20, "d"
50 PRINT TAB 26, "e"
```

sustituya el *punto y coma* del final por una *coma* y observe el nuevo resultado. Las cinco sentencias anteriores podrían editarse en una sola:

```
10 PRINT TAB 2, "a", TAB 8, "b", TAB 14, "c", TAB 20, "d", TAB 26, "e", TAB 32, "f"
```

AT

Acceso al teclado

CODE



IN


```

220 LET total=pato+gallina+cone
jo+paloma+cerdo
230 PRINT AT 15,6;"Total .....
";total
240 REM *****
* CALCULO PATAS *
*****
250 LET pat=pato*2
260 LET ga=gallina*2
270 LET co=conejo*4
280 LET pal=paloma*2
290 LET ce=cerdo*4
300 LET patas=pat+ga+co+pal+ce
400 PRINT AT 17,6;"Patras .....
";patas

```

PROGRAMA 2

```

10 REM *****
* CURSO BASIC *
*****
* ECUAC. 2 Gr *
*****
20 BORDER 1: PAPER 1: INK 7. C
LS
30 REM *****
* INTRODUCCION *
*****
40 PRINT " Este programa cal
cula las ra-ices de una ecuacion
de segundo grado."
50 PRINT "" La formula gene
ral de este tipo de ecuaciones e
s."
60 PRINT ""  $ax^2 + bx +$ 
 $c = 0$ "
70 PRINT "" De donde se ded
uce la incognita:"
80 PRINT ""
90 PRINT " 
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

100 PRINT "
110 PRINT ""
120 PRINT #0, " Pulse una tecl
a para seguir."
130 PAUSE 0
140 CLS
150 PRINT " El programa e
sta pidiendo el valor de "a".
"b" y "c".
160 PRINT " Posteriormente
realizara el calculo y seran
visualizadas las dos raices."
170 PRINT #0, " Pulse una tecla
para comenzar."

```

Canales de Comunicación

El Spectrum dispone de una serie de canales de comunicación o «streams», por los que el ordenador mantiene el intercambio de información con sus periféricos. Estos canales están numerados de «0» al «15» y precedidos por el signo del sostenido (#).

Pueden ser de entrada (INPUT) si los datos llegan al ordenador procedentes de alguno de sus periféricos, de salida (OUTPUT), si el ordenador es el que los envía y de entrada/salida (I/O), si el canal sirve tanto para enviar como para recibir datos.

Los canales «0» al «3» tienen una asignación fija de periféricos y el resto puede ser utilizado por cualquiera de ellos, previa definición por el usuario.

A través del canal # 2 se pueden visualizar valores numéricos o cadenas en la zona de pantalla destinada al usuario, por tanto es indiferente utilizar el comando «PRINT # 2» o «PRINT», ya que este último lleva asignado, implícitamente el canal de comunicación # 2.

El canal # es el asignado al periférico conocido como impresora, por tanto será de salida. Una instrucción del tipo:

```
PRINT # 3; "MICROHOBBY"
```

imprimirá la misma cadena que la sentencia específica de la impresora (LPRINT):

```
LPRINT "MICROHOBBY"
```

Los canales # 0 y # 1 son de entrada/salida y están relacionados con las dos últimas líneas de la pantalla, la 22 y 23. Estas líneas están destinadas para la introducción de sentencias en la edición de pro-


```

180 PAUSE 0
190 CLS
200 REM *****
* ENTRADA DE DATOS *
*
210 INPUT AT 0,0, "Valor de 'a' : ",a
    b,AT 1,0, "Valor de 'b' : ",b
    c,AT 2,0, "Valor de 'c' : ",c
220 REM *****
* VISUALIZACION *
*
230 PRINT "a = ",a
240 PRINT "b = ",b
250 PRINT "c = ",c
260 REM *****
* CALCULO DE RAICES *
*
270 LET raiz=SQR (b*b-(4*a*c))
280 LET divisor=2*a
290 LET raiz1=(-b+raiz)/divisor
300 LET raiz2=(-b-raiz)/divisor
400 REM *****
* VISUALIZACION DE *
*
410 PRINT ""Primera raiz ....
    " ;raiz1
420 PRINT ""Segunda raiz ....
    " ;raiz2

```

gramas, a la de datos cuando se utiliza la sentencia «INPUT» y para la visualización de los informes del ordenador.

Como ya vimos, con la sentencia PRINT no se podía visualizar ningún texto en estas dos líneas, sin embargo utilizando cualquiera de estos dos canales (0 ó 1) esto es posible. Ejecute las siguientes instrucciones y lo comprobará:

```

10 FOR I=0 TO 1
20 REM *****
30 PRINT AT I,0, "LINEA " ;I
40 INPUT AT I,0, " "

```

Al pulsar cualquier tecla, observará que las líneas 22 y

23 desaparecen, ¿por qué ocurre esto? Como hemos dicho anteriormente, estas líneas también las utiliza el ordenador para enviarnos sus mensajes, por tanto al visualizar el informe de fin de programa:

OK, 001

estas desaparecen.

Para aprovechar al máximo la capacidad de la zona de visualización destinada al usuario, los mensajes que tengan que enviarse en un programa, pueden hacerse a través de estos canales.

Ejemplos:

```

10 PRINT "OK"
20 FOR I=0 TO 1
30 PRINT AT I,0, " "
40 PAUSE 0

```

```

10 PRINT "OK"
20 FOR I=0 TO 1
30 PRINT AT I,0, " "
40 PAUSE 0

```

La instrucción «PAUSE 0» detiene la ejecución de un programa hasta que se pulsa una tecla, y la sentencia «BEEP 0,05,20» hace que suene el altavoz interno del Spectrum durante 0.05 sg. y con un tono de valor 20.

La impresión a través de los canales #0 y #1 puede combinarse con las sentencias auxiliares TAB y AT, teniendo en cuenta que para esta última, la línea 22 se convierte en la 0 y la 23 en la 1

Ejemplos:

```

10 PRINT AT 0,TAB 10, "OK"
20 PAUSE 0
30 BEEP 0,0,2

```

```

10 PRINT AT 0,0, "OK"
20 PRINT AT 1,0, " "
30 PRINT AT 0,0, " "
40 PAUSE 0

```

Cuando se utiliza un canal que no está activado, aparece el siguiente mensaje:

Invalid stream

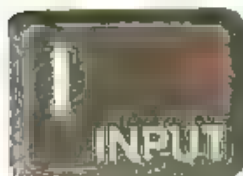
Al especificar una operación de entrada en un canal destinado a salida el informe de error es:

J Invalid I/O device

INPUT

Acceso al teclado

CODE



MODO **K**

IN

Tipo de sentencia

Comando de entrada.

Concepto

Esta sentencia permite introducir por teclado, durante la ejecución de un programa, datos, tanto numéricos como de cadena y se asignan a la variable indicadas en el argumento. Los datos introducidos se visualizan en las dos líneas inferiores de la pantalla, si cometemos algún error, podemos corregirlo con la función «DELETE». Para que los datos sean aceptados por el ordenador y continúe, por tanto, la ejecución del programa, debe pulsarse la tecla «ENTER».

Las estructuras básicas de esta sentencia son:

a)

SENTENCIA	ARGUMENTO
INPUT	Var. numérica

Ejemplos:

```
10 T=1: r=0
20 PRINT "r="
30 INPUT r
40 PRINT "r="
50 INPUT r
60 PRINT "r="
```

Si se introduce un valor no numérico, el intérprete de

PROGRAMA 3

```
10 REM *****
*      CURSO  BASICO      *
*      *****            *
*      INTERES SIMPLE      *
*      *****            *
11 BORDER 1: PAPER 1: INK 7: C
LS
12 REM *****
*      INTRODUCCION      *
*      *****            *
13 PRINT AT 2,3;"Este programa
calcula el inte"
14 PRINT AT 4,0;"res simple, de
acuerdo con la for"
15 PRINT AT 6,0;"mula."
16 PRINT AT 10,11;"C * R * T"
17 PRINT AT 11,7;"I = _____"
18 PRINT AT 13,14;100: PAUSE 3
00: C_L3
19 REM *****
*      ENTRADA DE      *
*      DATOS            *
*      *****            *
20 INPUT "CAPITAL: ";capital
30 PRINT "CAPITAL .... ";capit
al;" pesetas"
40 INPUT "REDITOS en % ";redi
tos
50 PRINT "REDITOS .... ";redi
tos;" % anual"
60 INPUT "TIEMPO: ";tiempo
70 PRINT "TIEMPO ..... ";tiem
po;" (anualidad)"
80 REM *****
*      CALCULO      *
*      *****            *
90 LET interes=capital*reditos
*tiempo/100
100 REM *****
*      RESULTADO      *
*      *****            *
110 PRINT "INTERESES .. ";inter
es;" pesetas"
```

Spectrum entenderá que es el nombre de una variable numérica, si esta no existe como tal, aparecerá el mensaje de error:

2 Variable not found

b)

SENTENCIA	ARGUMENTO
INPUT	Var. cadena

Ejemplos:

```
10 INPUT A$
20 PRINT A$
30 INPUT B$
40 PRINT B$
50 INPUT C$
60 PRINT C$
```

Observará que cuando la variable es del tipo alfanumérico o de cadena aparecen automáticamente las comillas en la zona destinada a la introducción de datos.

Dentro de la sentencia «INPUT» pueden especificarse más de una variable, éstas deben ir separadas por cualquiera de los signos ortográficos («», «;» o «,») con el mismo significado que con la sentencia «PRINT».

Ejemplos:

```
10 INPUT A$, B$, C$
20 PRINT A$, B$, C$
30 INPUT A$, B$, C$
40 PRINT A$, B$, C$
50 INPUT A$, B$, C$
60 PRINT A$, B$, C$
```

Otra posibilidad de la sentencia «INPUT» es la de presentar en pantalla un mensaje informativo, indicando qué tipo de dato debe introducir el usuario, este sistema es más eficaz ya que aclara las posibles dudas al respecto. E. mensaje, al ser alfanumérico, debe ir entrecomillado.

PROGRAMA 4

```
10 REM *****
*   CURSO BASIC   *
* *****        *
*   GRADOS        *
* *****        *
20 BORDER 1: PAPER 1: INK 7: C
LS
30 REM *****
*   INTRODUCCION  *
* *****        *
40 PRINT "   EL programa GRADO
S convierte""primera mente los
grados Centígrados""dos en Fahren
heit y posteriormen""te realiz
a la transformacion in-""versa
"
50 PRINT ""
60 PRINT ""          F = 9"
32"          * C +
70 PRINT ""          5"
60 PRINT ""          (F - 32
) * 5"
90 PRINT "          C =
100 PRINT ""          9"
110 PRINT #0;" Pulse una tecla
para seguir"
120 PAUSE 0
130 CLS
140 REM *****
*   ENTRADA       *
*   CENTIGRADOS   *
* *****        *
150 INPUT "Grados Centígrados?
", C
160 PRINT ""C;" grados Centígr
ados: "
170 REM *****
*   CALCULO "F"   *
* *****        *
180 LET fahrenheit=9*C/5+32
190 PRINT "igual a = "; fahrenheit, " F"
200 REM *****
*   ENTRADA       *
* *****        *
```

```

210 INPUT "Grados Fahrenheit? "
;f
220 PRINT "****f; " grados Fahren
heit. "
230 REM
      *****
      *
      * CALCULO "C" *
      *
      *****

240 LET centigrados=(f-32)*5/9
250 PRINT "Igual a = ";centigr
ados, " C"

```

PROGRAMA 5

```

10 REM *****
   * CURSO BASIC *
   *****
   *
   * FICHA *
   *
   *****

20 BORDER 1: PAPER 1: INK 6: C
-S
30 REM *****
   *
   * DIBUJO FICHA *
   *
   *****

50 LET a$="
60 LET b$="

70 PRINT "┌" + a$ + "┐"
80 FOR n=1 TO 20
90 PRINT AT n,0,"┌",AT n,31;"┐"

100 NEXT n
110 PRINT "└" + b$ + "┘"
120 REM
      *****
      *
      * ROTULOS FICHA *
      *
      *****

130 PRINT AT 2,3 "NOMBRE:"
140 PRINT AT 5,11;"APELLIDOS"
150 PRINT AT 6,1,
150 PRINT AT 8,3,"1 ";AT 8,16,"
2
170 PRINT AT 11,3,"CALLE ",AT 1
1,23;"N "
180 PRINT AT 13,3;"PISO " AT 1
3,16;"PUERTA:"

```

Ejemplos:

```

20 INPUT Nombre; "n$
30 INPUT P$; "apellido: "
40 INPUT "Segundo apellido: "
50 INPUT Ca$; "c$
60 INPUT Numero; "n
70 PRINT "Nombre: " + Ca$ + "
80 PRINT "Apellido: " + P$ + "
90 PRINT "Segundo apellido: "
100 PRINT " "

```

Cuando entre los comentarios del mensaje deba figurar el contenido de una variable, esta deberá ir encerrada entre paréntesis, bien ella sola, bien todo el mensaje, ya que de lo contrario, el intérprete BASIC la tomará como variable a introducir.

Ejemplos:

```

20 INPUT Nombre; "n$
30 INPUT "Mostrando en que
ca$; "c$
40 INPUT "En que número de la
50 PRINT "Te voy a mostrar
60 PRINT " "
70 PRINT " "

```

INPUT TAB y AT

Las sentencias auxiliares «TAB» y «AT» también pueden utilizarse conjuntamente con «INPUT». La palabra clave «TAB» se utiliza de forma similar que cuando acompaña a la sentencia «PRINT».

Ejemplos:

```

10 INPUT TAB 20;"Nombre: "
20 INPUT TAB 20;"Apellido: "
30 INPUT TAB 20;"Segundo apellido: "
40 INPUT TAB 20;"Número: "
50 INPUT TAB 20;"Dirección: "

```

«AT» tiene un tratamiento ligeramente distinto. Todos los «INPUT» a utilizar con «AT» deben estar incluidos en la misma instrucción, separados por «;». Independientemente de las coordenadas del primer «AT», la entrada de datos del primer «INPUT» se realiza en la zona inferior de la pantalla. Los siguientes se van a introducir en las coordenadas indicadas en los «AT», pero tomando como línea de referencia la especificada en el primer «AT».

```

190 PRINT AT 16,3,"TELEFONO "
200 PRINT AT 19,3,"POBLACION "
210 PRINT #0," Rellene los campos de la ficha"
220 PAUSE 200
230 REM

```

```

*****
*
* ENTRADA DE DATOS
*
*****

```

```

240 PRINT AT 2,11; FLASH 1; "<"
250 INPUT " "; LINE a$
260 PRINT AT 2,11,n$
270 PRINT AT 8,6; FLASH 1,"<"
280 INPUT " "; LINE a$
290 PRINT AT 8,6;a$
300 PRINT AT 8,19; FLASH 1,"<"
310 INPUT " "; LINE b$
320 PRINT AT 8,19;b$
330 PRINT AT 11,10; FLASH 1,"<"
340 INPUT " "; LINE c$
350 PRINT AT 11,10,c$
360 PRINT AT 11,26; FLASH 1,"<"
370 INPUT " "; nu
380 PRINT AT 11,26,nu
390 PRINT AT 13,10; FLASH 1,"<"
400 INPUT " "; pi
410 PRINT AT 13,10,pi
420 PRINT AT 13,26; FLASH 1,"<"
430 INPUT " "; d$
440 PRINT AT 13,26,d$
450 PRINT AT 16,14; FLASH 1,"<"
460 INPUT " "; te
470 PRINT AT 16,14,te
480 PRINT AT 19,14; FLASH 1,"<"
490 INPUT " "; LINE p$
500 PRINT AT 19,14,p$
510 PRINT #0,AT 1,10,"Fin de edición"
520 PAUSE 200

```

Veamos algunos ejemplos:

```

10 INPUT AT 0 0 Nombre a$
AT 1 0 Donde na ista a$
15 AT 3 0 A que provincia a ac
rtiene b$
20 AT 5 0 a Comunidad Aut
onoma d$
30 PRINT b$ se encuentra en
la p o y l i a de c y p e r i a
de a a p o b l a c i o n A u t o n o m a d e

```

```

10 INPUT AT 10 0 LINE a$
AT 2 0 LINE b$
15 AT 3 0 LINE c$
20 AT 4 0 LINE d$
25 AT 5 0 LINE e$
30 AT 6 0 LINE f$
35 AT 7 0 LINE g$
40 AT 8 0 LINE h$
45 AT 9 0 LINE i$
50 AT 10 0 LINE j$
55 AT 11 0 LINE k$
60 AT 12 0 LINE l$
65 AT 13 0 LINE m$
70 AT 14 0 LINE n$
75 AT 15 0 LINE o$
80 AT 16 0 LINE p$
85 AT 17 0 LINE q$
90 AT 18 0 LINE r$
95 AT 19 0 LINE s$
100 AT 20 0 LINE t$
105 AT 21 0 LINE u$
110 AT 22 0 LINE v$
115 AT 23 0 LINE w$
120 AT 24 0 LINE x$
125 AT 25 0 LINE y$
130 AT 26 0 LINE z$
135 AT 27 0 LINE aa$
140 AT 28 0 LINE ab$
145 AT 29 0 LINE ac$
150 AT 30 0 LINE ad$
155 AT 31 0 LINE ae$
160 AT 32 0 LINE af$
165 AT 33 0 LINE ag$
170 AT 34 0 LINE ah$
175 AT 35 0 LINE ai$
180 AT 36 0 LINE aj$
185 AT 37 0 LINE ak$
190 AT 38 0 LINE al$
195 AT 39 0 LINE am$
200 AT 40 0 LINE an$
205 AT 41 0 LINE ao$
210 AT 42 0 LINE ap$
215 AT 43 0 LINE aq$
220 AT 44 0 LINE ar$
225 AT 45 0 LINE as$
230 AT 46 0 LINE at$
235 AT 47 0 LINE au$
240 AT 48 0 LINE av$
245 AT 49 0 LINE aw$
250 AT 50 0 LINE ax$
255 AT 51 0 LINE ay$
260 AT 52 0 LINE az$
265 AT 53 0 LINE ba$
270 AT 54 0 LINE bb$
275 AT 55 0 LINE bc$
280 AT 56 0 LINE bd$
285 AT 57 0 LINE be$
290 AT 58 0 LINE bf$
295 AT 59 0 LINE bg$
300 AT 60 0 LINE bh$
305 AT 61 0 LINE bi$
310 AT 62 0 LINE bj$
315 AT 63 0 LINE bk$
320 AT 64 0 LINE bl$
325 AT 65 0 LINE bm$
330 AT 66 0 LINE bn$
335 AT 67 0 LINE bo$
340 AT 68 0 LINE bp$
345 AT 69 0 LINE bq$
350 AT 70 0 LINE br$
355 AT 71 0 LINE bs$
360 AT 72 0 LINE bt$
365 AT 73 0 LINE bu$
370 AT 74 0 LINE bv$
375 AT 75 0 LINE bw$
380 AT 76 0 LINE bx$
385 AT 77 0 LINE by$
390 AT 78 0 LINE bz$
395 AT 79 0 LINE ca$
400 AT 80 0 LINE cb$
405 AT 81 0 LINE cc$
410 AT 82 0 LINE cd$
415 AT 83 0 LINE ce$
420 AT 84 0 LINE cf$
425 AT 85 0 LINE cg$
430 AT 86 0 LINE ch$
435 AT 87 0 LINE ci$
440 AT 88 0 LINE cj$
445 AT 89 0 LINE ck$
450 AT 90 0 LINE cl$
455 AT 91 0 LINE cm$
460 AT 92 0 LINE cn$
465 AT 93 0 LINE co$
470 AT 94 0 LINE cp$
475 AT 95 0 LINE cq$
480 AT 96 0 LINE cr$
485 AT 97 0 LINE cs$
490 AT 98 0 LINE ct$
495 AT 99 0 LINE cu$
500 AT 100 0 LINE cv$
505 AT 101 0 LINE cw$
510 AT 102 0 LINE cx$
515 AT 103 0 LINE cy$
520 AT 104 0 LINE cz$
525 AT 105 0 LINE da$
530 AT 106 0 LINE db$
535 AT 107 0 LINE dc$
540 AT 108 0 LINE dd$
545 AT 109 0 LINE de$
550 AT 110 0 LINE df$
555 AT 111 0 LINE dg$
560 AT 112 0 LINE dh$
565 AT 113 0 LINE di$
570 AT 114 0 LINE dj$
575 AT 115 0 LINE dk$
580 AT 116 0 LINE dl$
585 AT 117 0 LINE dm$
590 AT 118 0 LINE dn$
595 AT 119 0 LINE do$
600 AT 120 0 LINE dp$
605 AT 121 0 LINE dq$
610 AT 122 0 LINE dr$
615 AT 123 0 LINE ds$
620 AT 124 0 LINE dt$
625 AT 125 0 LINE du$
630 AT 126 0 LINE dv$
635 AT 127 0 LINE dw$
640 AT 128 0 LINE dx$
645 AT 129 0 LINE dy$
650 AT 130 0 LINE dz$
655 AT 131 0 LINE ea$
660 AT 132 0 LINE eb$
665 AT 133 0 LINE ec$
670 AT 134 0 LINE ed$
675 AT 135 0 LINE ee$
680 AT 136 0 LINE ef$
685 AT 137 0 LINE eg$
690 AT 138 0 LINE eh$
695 AT 139 0 LINE ei$
700 AT 140 0 LINE ej$
705 AT 141 0 LINE ek$
710 AT 142 0 LINE el$
715 AT 143 0 LINE em$
720 AT 144 0 LINE en$
725 AT 145 0 LINE eo$
730 AT 146 0 LINE ep$
735 AT 147 0 LINE eq$
740 AT 148 0 LINE er$
745 AT 149 0 LINE es$
750 AT 150 0 LINE et$
755 AT 151 0 LINE eu$
760 AT 152 0 LINE ev$
765 AT 153 0 LINE ew$
770 AT 154 0 LINE ex$
775 AT 155 0 LINE ey$
780 AT 156 0 LINE ez$
785 AT 157 0 LINE fa$
790 AT 158 0 LINE fb$
795 AT 159 0 LINE fc$
800 AT 160 0 LINE fd$
805 AT 161 0 LINE fe$
810 AT 162 0 LINE ff$
815 AT 163 0 LINE fg$
820 AT 164 0 LINE fh$
825 AT 165 0 LINE fi$
830 AT 166 0 LINE fj$
835 AT 167 0 LINE fk$
840 AT 168 0 LINE fl$
845 AT 169 0 LINE fm$
850 AT 170 0 LINE fn$
855 AT 171 0 LINE fo$
860 AT 172 0 LINE fp$
865 AT 173 0 LINE fq$
870 AT 174 0 LINE fr$
875 AT 175 0 LINE fs$
880 AT 176 0 LINE ft$
885 AT 177 0 LINE fu$
890 AT 178 0 LINE fv$
895 AT 179 0 LINE fw$
900 AT 180 0 LINE fx$
905 AT 181 0 LINE fy$
910 AT 182 0 LINE fz$
915 AT 183 0 LINE ga$
920 AT 184 0 LINE gb$
925 AT 185 0 LINE gc$
930 AT 186 0 LINE gd$
935 AT 187 0 LINE ge$
940 AT 188 0 LINE gf$
945 AT 189 0 LINE gh$
950 AT 190 0 LINE gi$
955 AT 191 0 LINE gj$
960 AT 192 0 LINE gk$
965 AT 193 0 LINE gl$
970 AT 194 0 LINE gm$
975 AT 195 0 LINE gn$
980 AT 196 0 LINE go$
985 AT 197 0 LINE gp$
990 AT 198 0 LINE gq$
995 AT 199 0 LINE gr$
1000 AT 200 0 LINE gs$
1005 AT 201 0 LINE gt$
1010 AT 202 0 LINE gu$
1015 AT 203 0 LINE gv$
1020 AT 204 0 LINE gw$
1025 AT 205 0 LINE gx$
1030 AT 206 0 LINE gy$
1035 AT 207 0 LINE gz$
1040 AT 208 0 LINE ha$
1045 AT 209 0 LINE hb$
1050 AT 210 0 LINE hc$
1055 AT 211 0 LINE hd$
1060 AT 212 0 LINE he$
1065 AT 213 0 LINE hf$
1070 AT 214 0 LINE hg$
1075 AT 215 0 LINE hh$
1080 AT 216 0 LINE hi$
1085 AT 217 0 LINE hj$
1090 AT 218 0 LINE hk$
1095 AT 219 0 LINE hl$
1100 AT 220 0 LINE hm$
1105 AT 221 0 LINE hn$
1110 AT 222 0 LINE ho$
1115 AT 223 0 LINE hp$
1120 AT 224 0 LINE hq$
1125 AT 225 0 LINE hr$
1130 AT 226 0 LINE hs$
1135 AT 227 0 LINE ht$
1140 AT 228 0 LINE hu$
1145 AT 229 0 LINE hv$
1150 AT 230 0 LINE hw$
1155 AT 231 0 LINE hx$
1160 AT 232 0 LINE hy$
1165 AT 233 0 LINE hz$
1170 AT 234 0 LINE ia$
1175 AT 235 0 LINE ib$
1180 AT 236 0 LINE ic$
1185 AT 237 0 LINE id$
1190 AT 238 0 LINE ie$
1195 AT 239 0 LINE if$
1200 AT 240 0 LINE ig$
1205 AT 241 0 LINE ih$
1210 AT 242 0 LINE ii$
1215 AT 243 0 LINE ij$
1220 AT 244 0 LINE ik$
1225 AT 245 0 LINE il$
1230 AT 246 0 LINE im$
1235 AT 247 0 LINE in$
1240 AT 248 0 LINE io$
1245 AT 249 0 LINE ip$
1250 AT 250 0 LINE iq$
1255 AT 251 0 LINE ir$
1260 AT 252 0 LINE is$
1265 AT 253 0 LINE it$
1270 AT 254 0 LINE iu$
1275 AT 255 0 LINE iv$
1280 AT 256 0 LINE iw$
1285 AT 257 0 LINE ix$
1290 AT 258 0 LINE iy$
1295 AT 259 0 LINE iz$
1300 AT 260 0 LINE ja$
1305 AT 261 0 LINE jb$
1310 AT 262 0 LINE jc$
1315 AT 263 0 LINE jd$
1320 AT 264 0 LINE je$
1325 AT 265 0 LINE jf$
1330 AT 266 0 LINE jg$
1335 AT 267 0 LINE jh$
1340 AT 268 0 LINE ji$
1345 AT 269 0 LINE jj$
1350 AT 270 0 LINE jk$
1355 AT 271 0 LINE jl$
1360 AT 272 0 LINE jm$
1365 AT 273 0 LINE jn$
1370 AT 274 0 LINE jo$
1375 AT 275 0 LINE jp$
1380 AT 276 0 LINE jq$
1385 AT 277 0 LINE jr$
1390 AT 278 0 LINE js$
1395 AT 279 0 LINE jt$
1400 AT 280 0 LINE ju$
1405 AT 281 0 LINE jv$
1410 AT 282 0 LINE jw$
1415 AT 283 0 LINE jx$
1420 AT 284 0 LINE jy$
1425 AT 285 0 LINE jz$
1430 AT 286 0 LINE ka$
1435 AT 287 0 LINE kb$
1440 AT 288 0 LINE kc$
1445 AT 289 0 LINE kd$
1450 AT 290 0 LINE ke$
1455 AT 291 0 LINE kf$
1460 AT 292 0 LINE kg$
1465 AT 293 0 LINE kh$
1470 AT 294 0 LINE ki$
1475 AT 295 0 LINE kj$
1480 AT 296 0 LINE kl$
1485 AT 297 0 LINE km$
1490 AT 298 0 LINE kn$
1495 AT 299 0 LINE ko$
1500 AT 300 0 LINE kp$
1505 AT 301 0 LINE kq$
1510 AT 302 0 LINE kr$
1515 AT 303 0 LINE ks$
1520 AT 304 0 LINE kt$
1525 AT 305 0 LINE ku$
1530 AT 306 0 LINE kv$
1535 AT 307 0 LINE kw$
1540 AT 308 0 LINE kx$
1545 AT 309 0 LINE ky$
1550 AT 310 0 LINE kz$
1555 AT 311 0 LINE la$
1560 AT 312 0 LINE lb$
1565 AT 313 0 LINE lc$
1570 AT 314 0 LINE ld$
1575 AT 315 0 LINE le$
1580 AT 316 0 LINE lf$
1585 AT 317 0 LINE lg$
1590 AT 318 0 LINE lh$
1595 AT 319 0 LINE li$
1600 AT 320 0 LINE lj$
1605 AT 321 0 LINE lk$
1610 AT 322 0 LINE ll$
1615 AT 323 0 LINE lm$
1620 AT 324 0 LINE ln$
1625 AT 325 0 LINE lo$
1630 AT 326 0 LINE lp$
1635 AT 327 0 LINE lq$
1640 AT 328 0 LINE lr$
1645 AT 329 0 LINE ls$
1650 AT 330 0 LINE lt$
1655 AT 331 0 LINE lu$
1660 AT 332 0 LINE lv$
1665 AT 333 0 LINE lw$
1670 AT 334 0 LINE lx$
1675 AT 335 0 LINE ly$
1680 AT 336 0 LINE lz$
1685 AT 337 0 LINE ma$
1690 AT 338 0 LINE mb$
1695 AT 339 0 LINE mc$
1700 AT 340 0 LINE md$
1705 AT 341 0 LINE me$
1710 AT 342 0 LINE mf$
1715 AT 343 0 LINE mg$
1720 AT 344 0 LINE mh$
1725 AT 345 0 LINE mi$
1730 AT 346 0 LINE mj$
1735 AT 347 0 LINE mk$
1740 AT 348 0 LINE ml$
1745 AT 349 0 LINE mn$
1750 AT 350 0 LINE mo$
1755 AT 351 0 LINE mp$
1760 AT 352 0 LINE mq$
1765 AT 353 0 LINE mr$
1770 AT 354 0 LINE ms$
1775 AT 355 0 LINE mt$
1780 AT 356 0 LINE mu$
1785 AT 357 0 LINE mv$
1790 AT 358 0 LINE mw$
1795 AT 359 0 LINE mx$
1800 AT 360 0 LINE my$
1805 AT 361 0 LINE mz$
1810 AT 362 0 LINE na$
1815 AT 363 0 LINE nb$
1820 AT 364 0 LINE nc$
1825 AT 365 0 LINE nd$
1830 AT 366 0 LINE ne$
1835 AT 367 0 LINE nf$
1840 AT 368 0 LINE ng$
1845 AT 369 0 LINE nh$
1850 AT 370 0 LINE ni$
1855 AT 371 0 LINE nj$
1860 AT 372 0 LINE nk$
1865 AT 373 0 LINE nl$
1870 AT 374 0 LINE nm$
1875 AT 375 0 LINE no$
1880 AT 376 0 LINE np$
1885 AT 377 0 LINE nq$
1890 AT 378 0 LINE nr$
1895 AT 379 0 LINE ns$
1900 AT 380 0 LINE nt$
1905 AT 381 0 LINE nu$
1910 AT 382 0 LINE nv$
1915 AT 383 0 LINE nw$
1920 AT 384 0 LINE nx$
1925 AT 385 0 LINE ny$
1930 AT 386 0 LINE nz$
1935 AT 387 0 LINE oa$
1940 AT 388 0 LINE ob$
1945 AT 389 0 LINE oc$
1950 AT 390 0 LINE od$
1955 AT 391 0 LINE oe$
1960 AT 392 0 LINE of$
1965 AT 393 0 LINE og$
1970 AT 394 0 LINE oh$
1975 AT 395 0 LINE oi$
1980 AT 396 0 LINE oj$
1985 AT 397 0 LINE ok$
1990 AT 398 0 LINE ol$
1995 AT 399 0 LINE om$
2000 AT 400 0 LINE on$
2005 AT 401 0 LINE oo$
2010 AT 402 0 LINE op$
2015 AT 403 0 LINE oq$
2020 AT 404 0 LINE or$
2025 AT 405 0 LINE os$
2030 AT 406 0 LINE ot$
2035 AT 407 0 LINE ou$
2040 AT 408 0 LINE ov$
2045 AT 409 0 LINE ow$
2050 AT 410 0 LINE ox$
2055 AT 411 0 LINE oy$
2060 AT 412 0 LINE oz$
2065 AT 413 0 LINE pa$
2070 AT 414 0 LINE pb$
2075 AT 415 0 LINE pc$
2080 AT 416 0 LINE pd$
2085 AT 417 0 LINE pe$
2090 AT 418 0 LINE pf$
2095 AT 419 0 LINE pg$
2100 AT 420 0 LINE ph$
2105 AT 421 0 LINE pi$
2110 AT 422 0 LINE pj$
2115 AT 423 0 LINE pk$
2120 AT 424 0 LINE pl$
2125 AT 425 0 LINE pm$
2130 AT 426 0 LINE pn$
2135 AT 427 0 LINE po$
2140 AT 428 0 LINE pp$
2145 AT 429 0 LINE pq$
2150 AT 430 0 LINE pr$
2155 AT 431 0 LINE ps$
2160 AT 432 0 LINE pt$
2165 AT 433 0 LINE pu$
2170 AT 434 0 LINE pv$
2175 AT 435 0 LINE pw$
2180 AT 436 0 LINE px$
2185 AT 437 0 LINE py$
2190 AT 438 0 LINE pz$
2195 AT 439 0 LINE qa$
2200 AT 440 0 LINE qb$
2205 AT 441 0 LINE qc$
2210 AT 442 0 LINE qd$
2215 AT 443 0 LINE qe$
2220 AT 444 0 LINE qf$
2225 AT 445 0 LINE qg$
2230 AT 446 0 LINE qh$
2235 AT 447 0 LINE qi$
2240 AT 448 0 LINE qj$
2245 AT 449 0 LINE qk$
2250 AT 450 0 LINE ql$
2255 AT 451 0 LINE qm$
2260 AT 452 0 LINE qn$
2265 AT 453 0 LINE qo$
2270 AT 454 0 LINE qp$
2275 AT 455 0 LINE qq$
2280 AT 456 0 LINE qr$
2285 AT 457 0 LINE qs$
2290 AT 458 0 LINE qt$
2295 AT 459 0 LINE qu$
2300 AT 460 0 LINE qv$
2305 AT 461 0 LINE qw$
2310 AT 462 0 LINE qx$
2315 AT 463 0 LINE qy$
2320 AT 464 0 LINE qz$
2325 AT 465 0 LINE ra$
2330 AT 466 0 LINE rb$
2335 AT 467 0 LINE rc$
2340 AT 468 0 LINE rd$
2345 AT 469 0 LINE re$
2350 AT 470 0 LINE rf$
2355 AT 471 0 LINE rg$
2360 AT 472 0 LINE rh$
2365 AT 473 0 LINE ri$
2370 AT 474 0 LINE rj$
2375 AT 475 0 LINE rk$
2380 AT 476 0 LINE rl$
2385 AT 477 0 LINE rm$
2390 AT 478 0 LINE rn$
2395 AT 479 0 LINE ro$
2400 AT 480 0 LINE rp$
2405 AT 481 0 LINE rq$
2410 AT 482 0 LINE rr$
2415 AT 483 0 LINE rs$
2420 AT 484 0 LINE rt$
2425 AT 485 0 LINE ru$
2430 AT 486 0 LINE rv$
2435 AT 487 0 LINE rw$
2440 AT 488 0 LINE rx$
2445 AT 489 0 LINE ry$
2450 AT 490 0 LINE rz$
2455 AT 491 0 LINE sa$
2460 AT 492 0 LINE sb$
2465 AT 493 0 LINE sc$
2470 AT 494 0 LINE sd$
2475 AT 495 0 LINE se$
2480 AT 496 0 LINE sf$
2485 AT 497 0 LINE sg$
2490 AT 498 0 LINE sh$
2495 AT 499 0 LINE si$
2500 AT 500 0 LINE sj$
2505 AT 501 0 LINE sk$
2510 AT 502 0 LINE sl$
2515 AT 503 0 LINE sm$
2520 AT 504 0 LINE sn$
2525 AT 505 0 LINE so$
2530 AT 506 0 LINE sp$
2535 AT 507 0 LINE sq$
2540 AT 508 0 LINE sr$
2545 AT 509 0 LINE ss$
2550 AT 510 0 LINE st$
2555 AT 511 0 LINE su$
2560 AT 512 0 LINE sv$
2565 AT 513 0 LINE sw$
2570 AT 514 0 LINE sx$
2575 AT 515 0 LINE sy$
2580 AT 516 0 LINE sz$
2585 AT 517 0 LINE ta$
2590 AT 518 0 LINE tb$
2595 AT 519 0 LINE tc$
2600 AT 520 0 LINE td$
2605 AT 521 0 LINE te$
2610 AT 522 0 LINE tf$
2615 AT 523 0 LINE tg$
2620 AT 524 0 LINE th$
2625 AT 525 0 LINE ti$
2630 AT 526 0 LINE tj$
2635 AT 527 0 LINE tk$
2640 AT 528 0 LINE tl$
2645 AT 529 0 LINE tm$
2650 AT 530 0 LINE tn$
2655 AT 531 0 LINE to$
2660 AT 532 0 LINE tp$
2665 AT 533 0 LINE tq$
2670 AT 534 0 LINE tr$
2675 AT 535 0 LINE ts$
2680 AT 536 0 LINE tt$
2685 AT 537 0 LINE tu$
2690 AT 538 0 LINE tv$
2695 AT 539 0 LINE tw$
2700 AT 540 0 LINE tx$
2705 AT 541 0 LINE ty$
2710 AT 542 0 LINE tz$
2715 AT 543 0 LINE ua$
2720 AT 544 0 LINE ub$
2725 AT 545 0 LINE uc$
2730 AT 546 0 LINE ud$
2735 AT 547 0 LINE ue$
2740 AT 548 0 LINE uf$
2745 AT 549 0 LINE ug$
2750 AT 550 0 LINE uh$
2755 AT 551 0 LINE ui$
2760 AT 552 0 LINE uj$
2765 AT 553 0 LINE uk$
2770 AT 554 0 LINE ul$
2775 AT 555 0 LINE um$
2780 AT 556 0 LINE un$
2785 AT 557 0 LINE uo$
2790 AT 558 0 LINE up$
2795 AT 559 0 LINE uq$
2800 AT 560 0 LINE ur$
2805 AT 561 0 LINE us$
2810 AT 562 0 LINE ut$
2815 AT 563 0 LINE uu$
2820 AT 564 0 LINE uv$
2825 AT 565 0 LINE uw$
2830 AT 566 0 LINE ux$
2835 AT 567 0 LINE uy$
2840 AT 568 0 LINE uz$
2845 AT 569 0 LINE va$
2850 AT 570 0 LINE vb$
2855 AT 571 0 LINE vc$
2860 AT 572 0 LINE vd$
2865 AT 573 0 LINE ve$
2870 AT 574 0 LINE vf$
2875 AT 575 0 LINE vg$
2880 AT 576 0 LINE vh$
2885 AT 577 0 LINE vi$
2890 AT 578 0 LINE vj$
2895 AT 579 0 LINE vk$
2900 AT 580 0 LINE vl$
2905 AT 581 0 LINE vm$
2910 AT 582 0 LINE vn$
2915 AT 583 0 LINE vo$
2920 AT 584 0 LINE vp$
2925 AT 585 0 LINE vq$
2930 AT 586 0 LINE vr$
2935 AT 587 0 LINE vs$
2940 AT 588 0 LINE vt$
2945 AT 589 0 LINE vu$
2950 AT 590 0 LINE vv$
2955 AT 591 0 LINE vw$
2960 AT 592 0 LINE vx$
2965 AT 593 0 LINE vy$
2970 AT 594 0 LINE vz$
2975 AT 595 0 LINE wa$
2980 AT 596 0 LINE wb$
2985 AT 597 0 LINE wc$
2990 AT 598 0 LINE wd$
2995 AT 599 0 LINE we$
3000 AT 600 0 LINE wf$
3005 AT 601 0 LINE wg$
3010 AT 602 0 LINE wh$
3015 AT 603 0 LINE wi$
3020 AT 604 0 LINE wj$
3025 AT 605 0 LINE wk$
3030 AT 606 0 LINE wl$
3035 AT 607 0 LINE wm$
3040 AT 608 0 LINE wn$
3045 AT 609 0 LINE wo$
3050 AT 610 0 LINE wp$
3055 AT 611 0 LINE wq$
3060 AT 612 0 LINE wr$
3065 AT 613 0 LINE ws$
3070 AT 614 0 LINE wt$
3075 AT 615 0 LINE wu$
3080 AT 616 0 LINE wv$
3085 AT 617 0 LINE ww$
3090 AT 618 0 LINE wx$
3095 AT 619 0 LINE wy$
3100 AT 620 0 LINE wz$
3105 AT 621 0 LINE xa$
3110 AT 622 0 LINE xb$
3115 AT 623 0 LINE xc$
3120 AT 624 0 LINE xd$
3125 AT 625 0 LINE xe$
3130 AT 626 0 LINE xf$
3135 AT 627 0 LINE xg$
3140 AT 628 0 LINE xh$
3145 AT 629 0 LINE xi$
3150 AT 630 0 LINE xj$
3155 AT 631 0 LINE xk$
3160 AT 632 0 LINE xl$
3165 AT 633 0 LINE xm$
3170 AT 634 0 LINE xn$
3175 AT 635 0 LINE xo$
3180 AT 636 0 LINE xp$
3185 AT 637 0 LINE xq$
3190 AT 638 0 LINE xr$
3195 AT 639 0 LINE xs$
3200 AT 640 0 LINE xt$
3205 AT 641 0 LINE xu$
3210 AT 642 0 LINE xv$
3215 AT 643 0 LINE xw$
3220 AT 644 0 LINE xx$
3225 AT 645 0 LINE xy$
3230 AT 646 0 LINE xz$
3235 AT 647 0 LINE ya$
3240 AT 648 0 LINE yb$
3245 AT 649 0 LINE yc$
3250 AT 650 0 LINE yd$
3255 AT 651 0 LINE ye$
3260 AT 652 0 LINE yf$
3265 AT 653 0 LINE yg$
3270 AT 654 0 LINE yh$
3275 AT 655 0 LINE yi$
3280 AT 656 0 LINE yj$
3285 AT 657 0 LINE yk$
3290 AT 658 0 LINE yl$
3295 AT 659 0 LINE ym$
3300 AT 660 0 LINE yn$
3305 AT 661 0 LINE yo$
3310 AT 662 0 LINE yp$
3315 AT 663 0 LINE yq$
3320 AT 664 0 LINE yr$
3325 AT 665 0 LINE ys$
3330 AT 666 0 LINE yt$
3335 AT 667 0 LINE yu$
3340 AT 668 0 LINE yv$
3345 AT 669 0 LINE yw$
3350 AT 670 0 LINE yx$
3355 AT 671 0 LINE yy$
3360 AT 672 0 LINE yz$
3365 AT 673 0 LINE za$
3370 AT 674 0 LINE zb$
3375 AT 675 0 LINE zc$
3380 AT 676 0 LINE zd$
3385 AT 677 0 LINE ze$
3390 AT 678 0 LINE zf$
3395 AT 679 0 LINE zg$
3400 AT 680 0 LINE zh$
3405 AT 681 0 LINE zi$
3410 AT 682 0 LINE zj$
3415 AT 683 0 LINE zk$
3420 AT 684 0 LINE zl$
3425 AT 685 0 LINE zm$
3430 AT 686 0 LINE zn$
3435 AT 687 0 LINE zo$
3440 AT 688 0 LINE zp$
3445 AT 689 0 LINE zq$
3450 AT 690 0 LINE zr$
3455 AT 691 0 LINE zs$
3460 AT 692 0 LINE zt$
3465 AT 693 0 LINE zu$
3470 AT 694 0 LINE zv$
3475 AT 695 0 LINE zw$
3480 AT 696 0 LINE zx$
3485 AT 697 0 LINE zy$
3490 AT 698 0 LINE zz$
3495 AT 699 0 LINE aa$
3500 AT 700 0 LINE ab$
3505 AT 701 0 LINE ac$
3510 AT 702 0 LINE ad$
3515 AT 703 0 LINE ae$
3520 AT 704 0 LINE af$
3525 AT 705 0 LINE ag$
3530 AT 706 0 LINE ah$
3535 AT 707 0 LINE ai$
3540 AT 708 0 LINE aj$
3545 AT 709 0 LINE ak$
3550 AT 710 0 LINE al$
3555 AT 711 0 LINE am$
3560 AT 712 0 LINE an$
3565 AT 713 0 LINE ao$
3570 AT 714 0 LINE ap$
3575 AT 715 0 LINE aq$
3580 AT 716 0 LINE ar$
3585 AT 717 0 LINE as$
3590 AT 718 0 LINE at$
3595 AT 719 0 LINE au$
3600 AT 720 0 LINE av$
3605 AT 721 0 LINE aw$
3610 AT 722 0 LINE ax$
3615 AT 723 0 LINE ay$
3620 AT 724 0 LINE az$
3625 AT 725 0 LINE ba$
3630 AT 726 0 LINE bb$
3635 AT 727 0 LINE bc$
3640 AT 728 0 LINE bd$
3645 AT 729 0 LINE be$
3650 AT 730 0 LINE bf$
3655 AT 731 0 LINE bg$
3660 AT 732 0 LINE bh$
3665 AT 733 0 LINE bi$
3670 AT 734 0 LINE bj$
3675 AT 735 0 LINE bk$
3680 AT 736 0 LINE bl$
3685 AT 737 0 LINE bm$
3690 AT 738 0 LINE bn$
3695 AT 739 0 LINE bo$
3700 AT 740 0 LINE bp$
3705 AT 741 0 LINE bq$
3710 AT 742 0 LINE br$
3715 AT 743 0 LINE bs$
3720 AT 744 0 LINE bt$
3725 AT 745 0 LINE bu$
3730 AT 746 0 LINE bv$
3735 AT 747 0 LINE bw$
3740 AT 748 0 LINE bx$
3745 AT 749 0 LINE by$
3750 AT 750 0 LINE bz$
3755 AT 751 0 LINE ca$
3760 AT 752 0 LINE cb$
3765 AT 753 0 LINE cc$
3770 AT 754 0 LINE cd$
3775 AT 755 0 LINE ce$
3780 AT 756 0 LINE cf$
3785 AT 757 0 LINE cg$
3790 AT 758 0 LINE ch$
3795 AT 759 0 LINE ci$
3800 AT 760 0 LINE cj$
3805 AT 761 0 LINE ck$
3810 AT 762 0 LINE cl$
3815 AT 763 0 LINE cm$
3820 AT 764 0 LINE cn$
3825 AT 765 0 LINE co$
3830 AT 766 0 LINE cp$
3835 AT 767 0 LINE cq$
3840 AT 768 0 LINE cr$
3845 AT 769 0 LINE cs$
3850 AT 770 0 LINE ct$
3855 AT 771 0 LINE cu$
3860 AT 772 0 LINE cv$
3865 AT 773 0 LINE cw$
3870 AT 774 0 LINE cx$
3875 AT 775 0 LINE cy$
3880 AT 776 0 LINE cz$
3885 AT 777 0 LINE da$
3890 AT 778 0 LINE db$
3895 AT 779 0 LINE dc$
3900 AT 780 0 LINE dd$
3905 AT 781 0 LINE de$
3910 AT 782 0 LINE df$
3915 AT 783 0 LINE dg$
3920 AT 784 0 LINE dh$
3925 AT 785 0 LINE di$
3930 AT 786 0 LINE dj$
3935 AT 787 0 LINE dk$
3940 AT 788 0 LINE dl$
3945 AT
```


Programas de repaso

Como colofón al estudio de las sentencias básicas de programación.

```
REM  
LET  
PRINT  
INPUT
```

se analizan en este capítulo cinco programas realizados con este tipo de instrucciones.

Estos programas, numerados del uno al cinco, son los siguientes:

```
1 GRANJA  
2 ECUACION  
3 INTERFS  
4 GRADOS  
5 FCHA
```

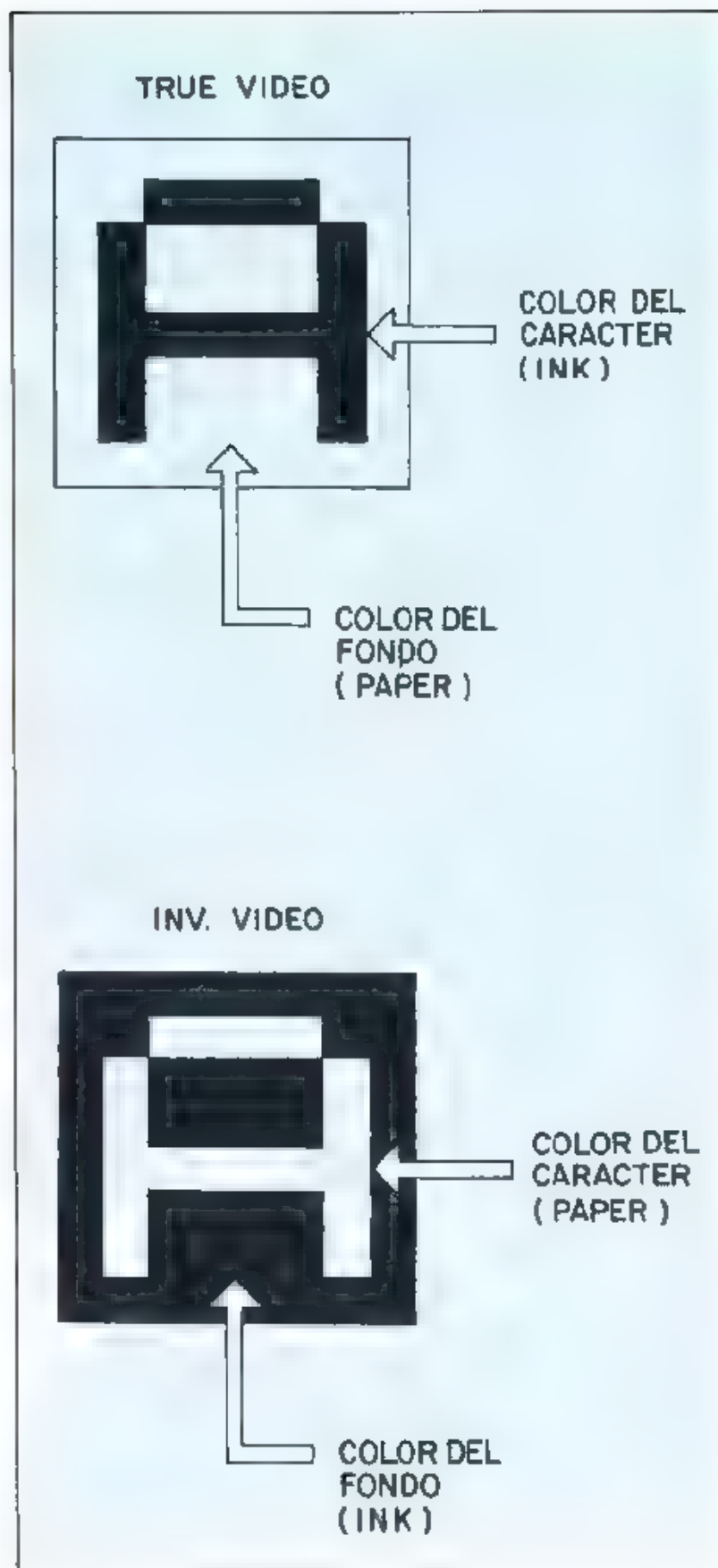
Programa «GRANJA»

Para almacenarlo en cinta una vez editado hágalo, por ejemplo, de la forma:

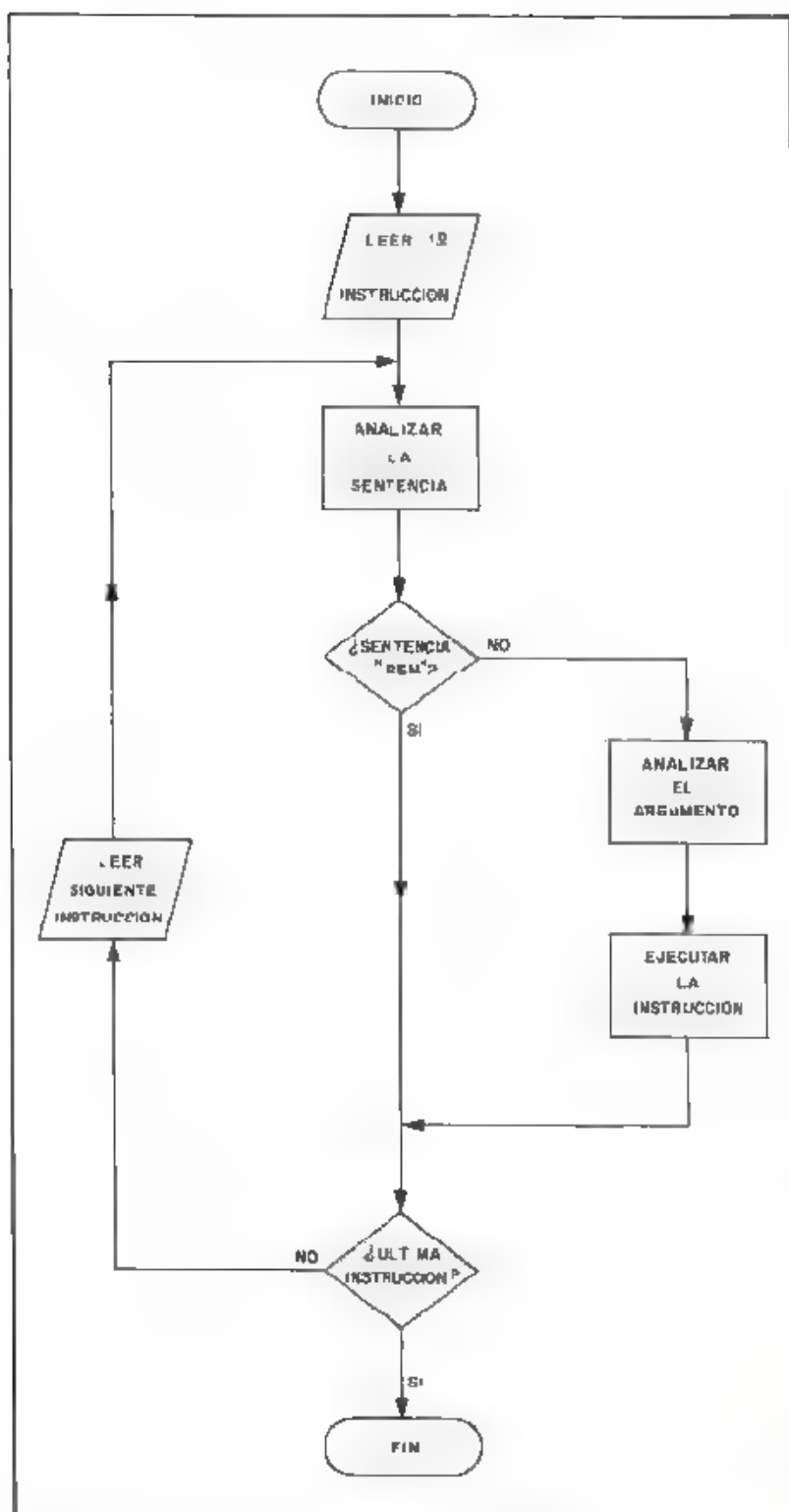
```
SAVE "granja"
```

Este programa es bastante sencillo, ya que calcula el número total de animales que hay en una granja, a partir de los datos que le son introducidos por teclado; también calcula el número total de patas.

La sentencia 10 es la presentación del programa. En la 20 hay cuatro sentencias, esto es posible ya que se utiliza el signo separador «:»; estas no se han explicado todavía, pero vamos a ver unas pequeñas nociones sobre su funcionamiento. En conjunto realizan la tarea de colorear la pantalla. La sentencia «BORDER 1» asigna el color azul al borde de la pantalla, «PAPER 1» asigna el color azul al fondo, e «INK 7» tiene la misión de que el color de los caracteres sea



Funciones de video.



Análisis sentencia "REM".

blanco. Por último, «CLS» es una sentencia que borra la pantalla y asigna de inmediato los colores especificados en los comandos anteriores.

Las sentencias 40 y 50 son

del tipo «PRINT AT» y sirven para dar al usuario una pequeña información sobre la finalidad del programa.

La instrucción 60 utiliza el canal de comunicación 0

para visualizar el mensaje de espera

«PAUSE 0» situada en la línea 70 es una instrucción que detiene la ejecución del programa hasta que se pulsa una tecla

La sentencia 80 borra la información visualizada en la pantalla.

De la línea 100 a la 140 se encuentran los «INPUT» necesarios para la entrada de datos. La variable numérica utilizada para almacenar el número total de patos es «pato», para los restantes animales se han utilizado: «gallina», «conejo», «paloma» y «cerdo»

Después de la introducción de datos, se realiza la visualización detallada de éstos, los «PRINT AT» de las líneas 160 a 200, se encargan de ello.

El cálculo del número total de animales se realiza en la línea 220, se asigna a la variable «total» la suma de las variables «pato», «gallina», «conejo», «paloma» y «cerdo». La línea 230 se encarga de visualizar este valor

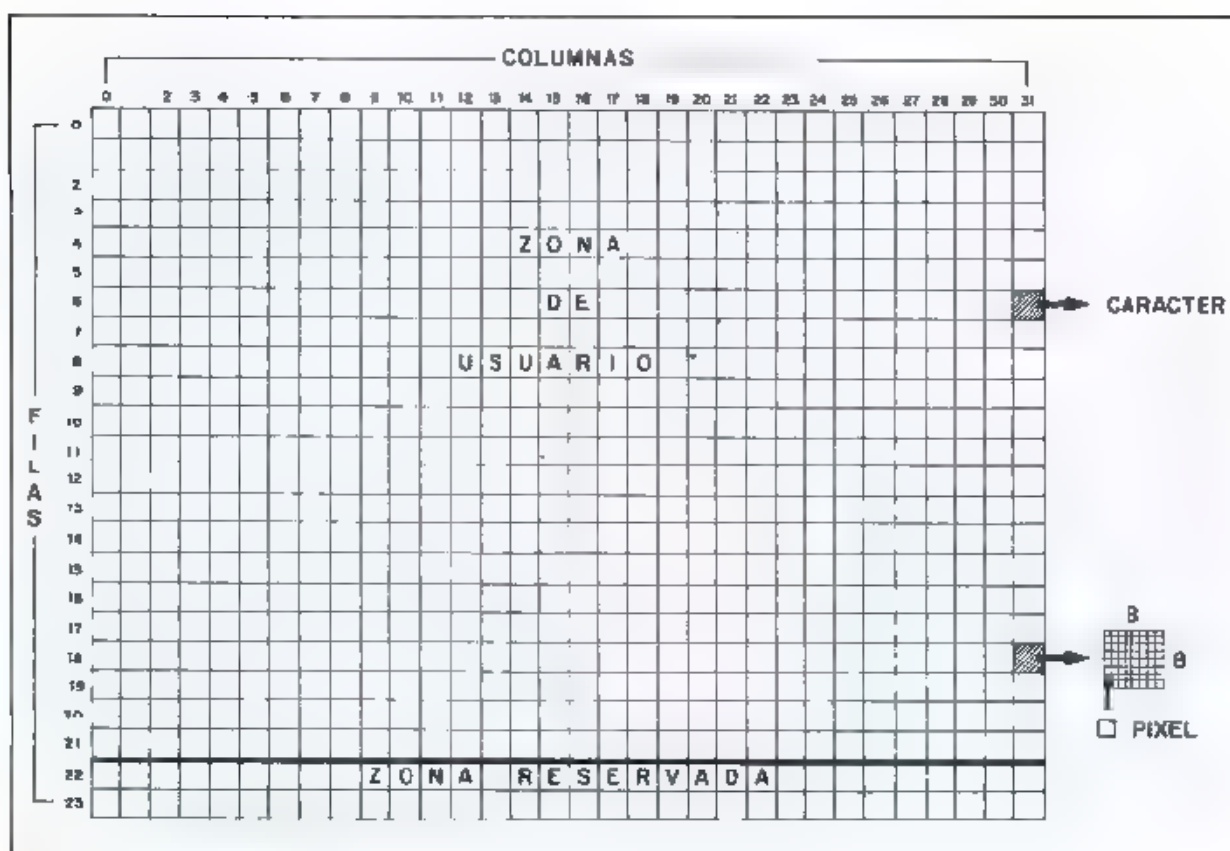
Las líneas 250 a 300 se encargan de calcular el número total de patas. Primero se asignan a las variables «pat», «gat», «co», «pal» y «ce» los valores totales por especie.

Ejemplo: como los conejos tiene cuatro patas, será necesario multiplicar este numero por el numero total de conejos, valor especificado en la variable «conejo»;

```
LET co = conejo * 4
```

Posteriormente se asigna a la variable «patas» la suma de las variables «pat», «ga», «co», «pal» y «ce».

Por último el valor de la variable «patas» se visualiza con el «PRINT AT» de la línea 400.



Zonas de visualización.

Programa «ECUACION»

Salvar el programa de la forma

SAVE "ecuación"

Este programa calcula las dos raíces de una ecuación de segundo grado del tipo:

$$ax^2 + bx + c = 0$$

Los dos valores de «x» que cumplen esta ecuación se calculan con la fórmula.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Los valores que deben introducirse para que el programa calcule dichas ecuaciones son los correspondientes a las variables «a», «b» y «c».

Las funciones de las sentencias que componen dicho programa son las siguientes.

- 1φ : Comentario con el nombre del programa.
- 2φ : Asignación de los colores de borde, papel y tinta.
- 4φ — 14φ : Primera pantalla de información.
- 15φ — 19φ : Segunda pantalla de información.
- 21φ : Introducción de los valores de las variables «a», «b» y «c».
- 23φ — 25φ : Visualización detallada de las variables.
- 27φ — 30φ : Cálculo de las dos raíces. En este programa se ha utilizado la sentencia «SQR» que calcula a raíz cua-

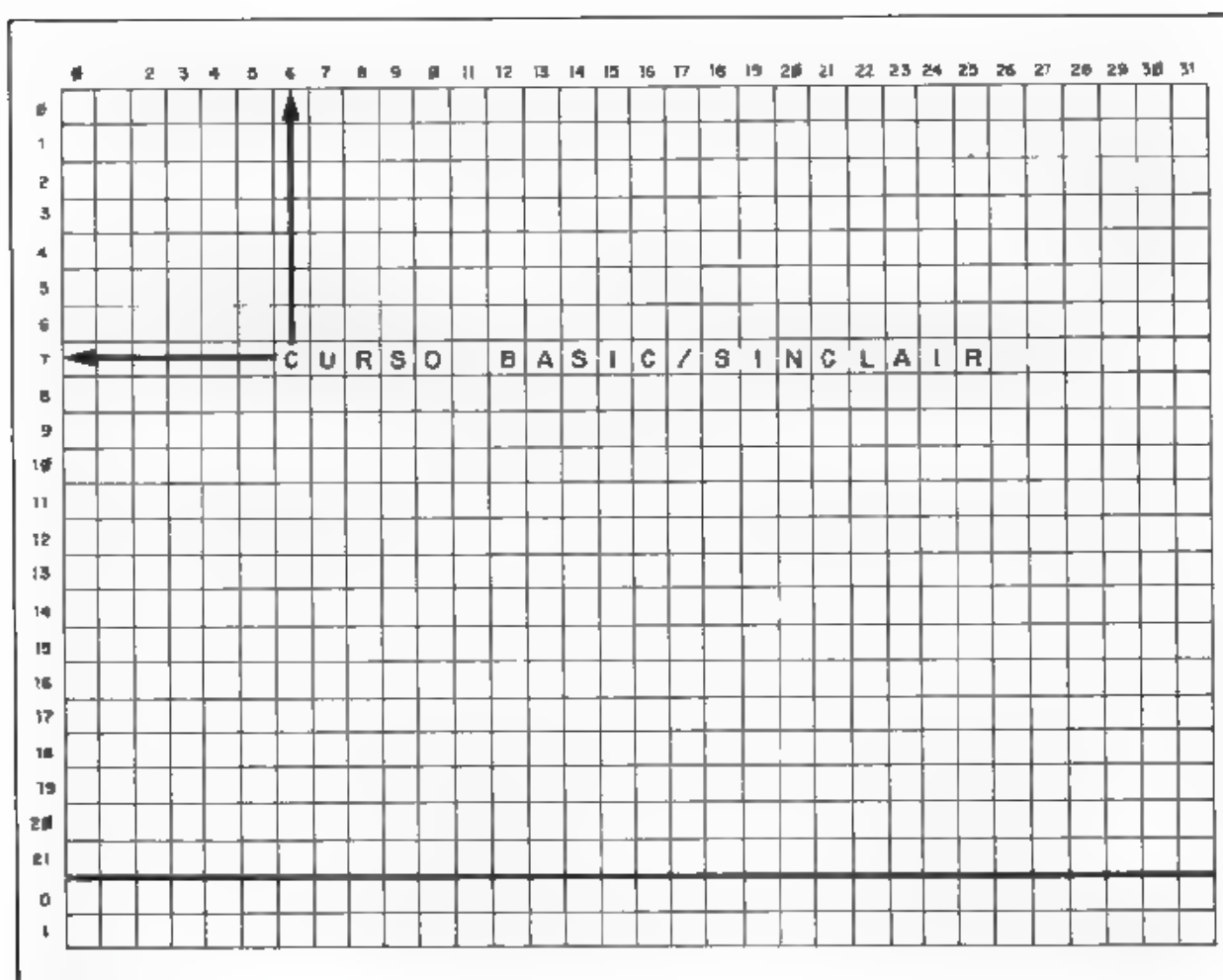
drada del argumento que va entre paréntesis. El cálculo se ha realizado en varias etapas, primeramente se han evaluado las partes comunes; a la variable «raíz» se le ha asignado el resultado de:

$$\sqrt{b^2 - 4ac}$$

y a la variable «divisor»

$$2a$$

posteriormente y a partir de estas dos variables se han obtenido las dos



Print AT 7,6.

raíces, «raíz 1» y «raíz 2»
 41Ø · 42Ø : Visualización de los resultados.

Programa «INTERES»

Grábalo en cinta de la forma.

SAVE "nterés"

Este programa calcula el *interés simple* de un capital colocado en un banco durante cierto número de años. La fórmula del interés simple implementada en el programa es:

$$I = \frac{C \cdot R \cdot T}{100}$$

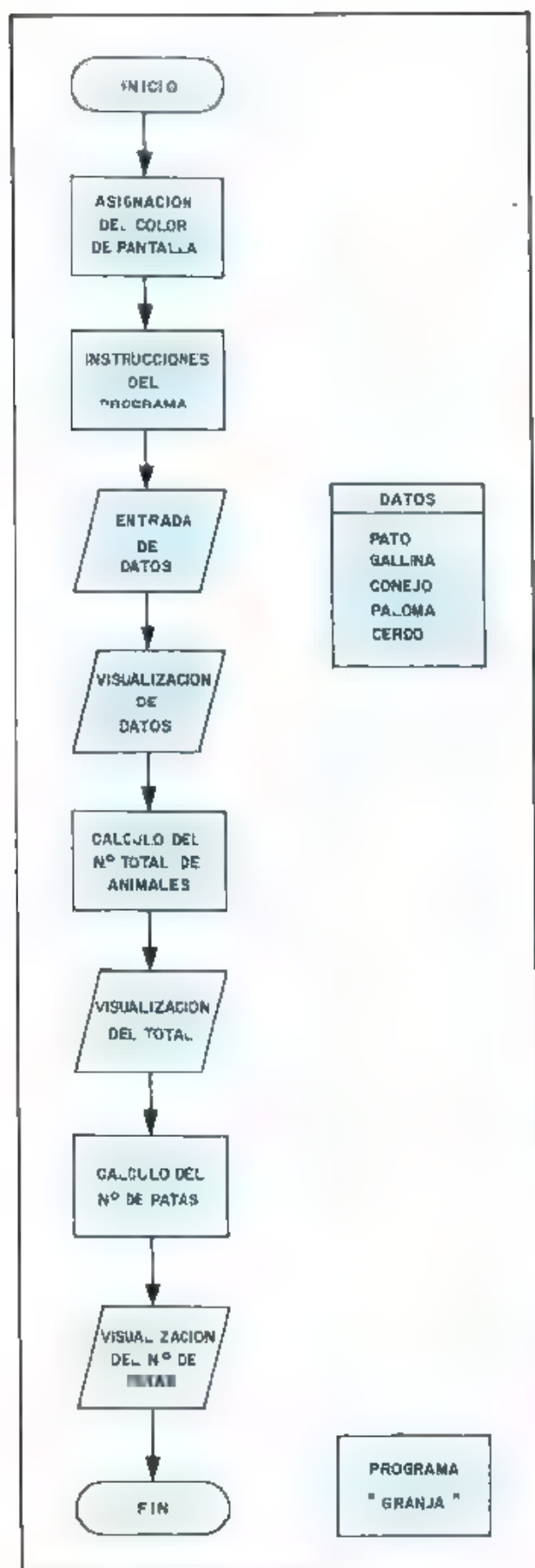
El valor asignado a la variable «capital» debe estar expresado en pesetas, el de la variable «reditos» en %, es decir, si el banco proporciona unos intereses al 3%, el valor a introducir deberá ser «3», y por último el asignado a la variable «tiempo» deberá ser expresado en años.

La estructura del programa es la siguiente:

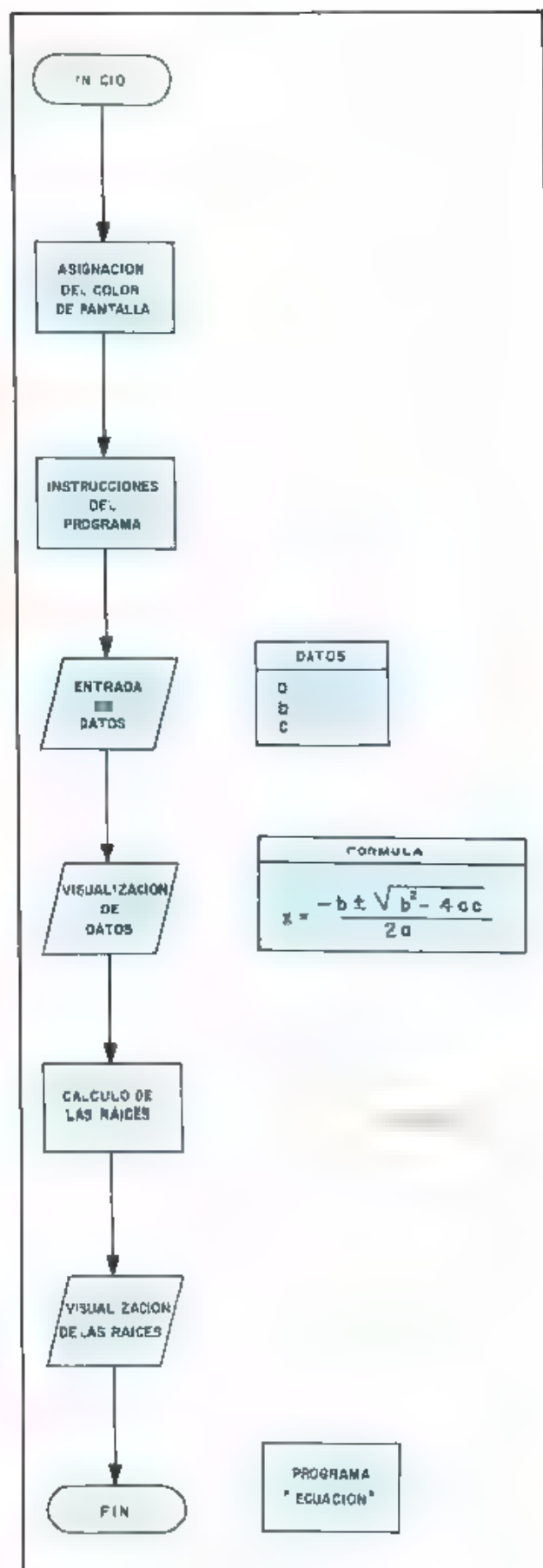
La estructura del programa es la siguiente:

- 1Ø : Comentario con el nombre del programa.
- 11 : Asignación del color azul para el borde y el fondo y blanco para los caracteres.
- 13 — 18 : Visualización de una breve descripción del

programa. La segunda sentencia de la línea 18 es «PAUSE 3ØØ», esta proporciona una temporización de aproximadamente seis segundos, desde que aparece la información hasta que se borra con la siguiente sentencia (CLS). Si durante la temporización se pulsa una tecla, ésta termina y se ejecuta la instrucción siguiente.



Programa "Granja".



Programa "Ecuación".



- 2φ — 7φ : Entrada y visualización del contenido de las variables «capital», «réditos» y «tiempo».
- 9φ : Cálculo de los intereses. A la variable «interés» se le asigna el resultado de la fórmula.
- 11φ : Visualización de los resultados.

Programa «GRADOS»

Almacenarlo en cinta, por ejemplo, de la forma:

SAVE "grados"

El programa «GRADOS» consta de dos partes, en la primera transforma un valor de grados centígrados (°C), introducido por teclado, en grados Fahrenheit (°F) de acuerdo con la fórmula.

$$^{\circ}\text{F} = \frac{9}{5} ^{\circ}\text{C} + 32$$

La variable «c» contiene los grados centígrados a transformar y la variable «fahrenheit» el resultado.

En la segunda parte hace la transformación inversa, es decir, transforma un valor de grados Fahrenheit en centígrados, la fórmula implementada en este caso es:

$$^{\circ}\text{C} = (^{\circ}\text{F} - 32) \cdot \frac{5}{9}$$

La variable «f» contiene los grados fahrenheit a transformar y la variable «centígrados» el resultado.

El programa ha sido estructurado de la siguiente manera:

- 1φ : Comentario con el nombre del programa
- 2φ : Asignación del color azul para el borde y el fondo y blanco

para los caracteres.

- 4φ — 13φ : Breve descripción del programa. En la línea 11φ se utiliza el canal de comunicación φ.

- 15φ — 16φ: Entrada y visualización de la variable «c»

- 18φ — 19φ: Cálculo y visualización del resultado en grados Fahrenheit.

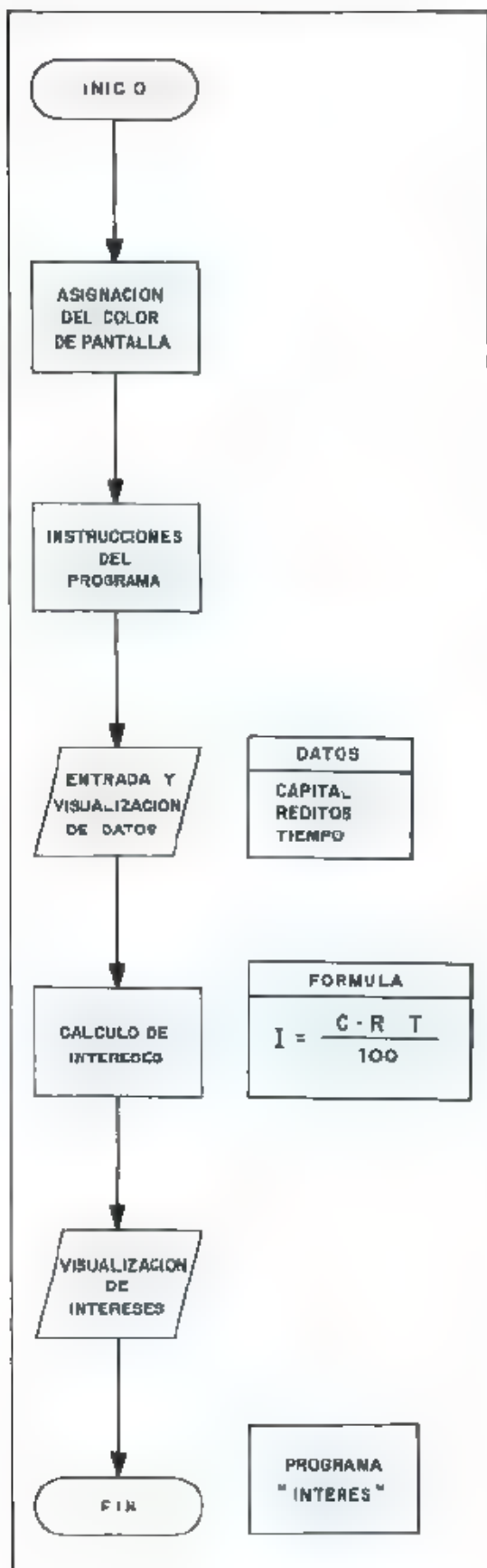
- 21φ — 22φ: Entrada y visualización de la variable «f»

- 24φ — 25φ: Cálculo y visualización del resultado en grados centígrados.

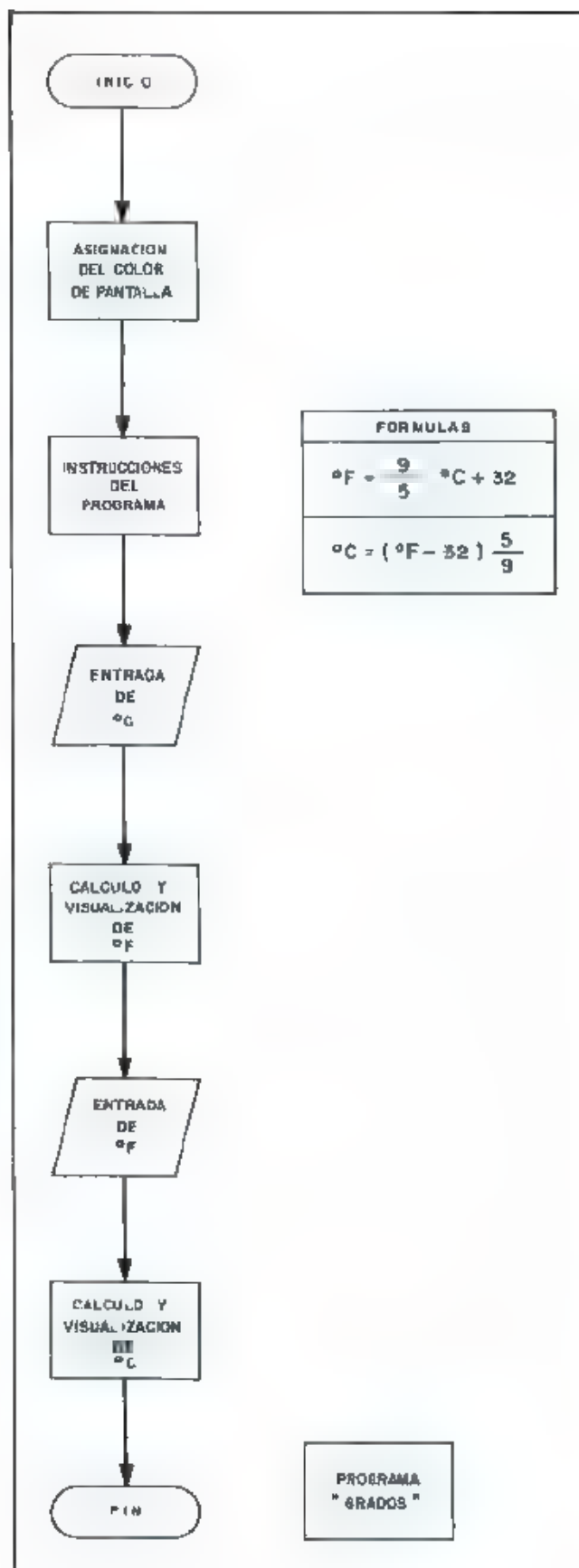
Programa «FICHA»

Salvarlo de la forma habitual:

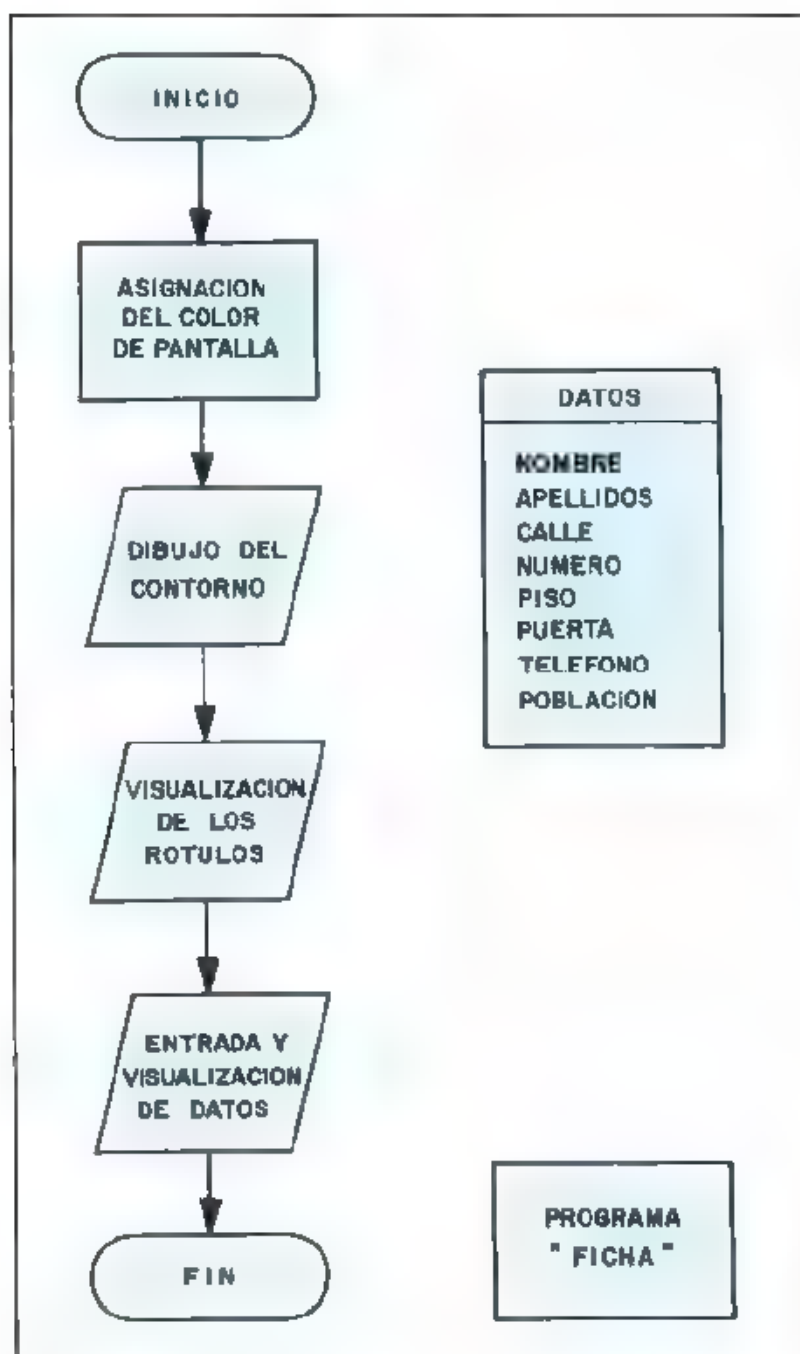
SAVE "ficha"



Programa "Interés".



Programa "Grados".



Programa "Ficha".

Este programa simula una ficha con diversos *campos* (nombre, apellidos, etc...) que el usuario debe rellenar. Este programa puede formar parte de otro mayor que almacene, por ejemplo, los datos de los empleados de una empresa, los datos de los clientes o los suministradores.

Al editar el programa, debe poner atención en las líneas

50 a 110, ya que van incluidos algunos de los gráficos predefinidos que incorpora el Spectrum. Debe pasar a modo **G** (gráficos).

Las líneas 80 a 100 forman lo que en programación se llama *bucle*, por tanto la instrucción 90 se repetirá 20 veces pero con diversos valores de la variable «n», ya que el índice del bucle está comprendido

entre los márgenes 1 y 20. Los bucles «FOR»...«NEXT» serán estudiados con mayor detalle en otro capítulo.

La sentencia «FLASH 1» antepuesta al símbolo «<» (menor que) hace que este parpadee en la pantalla para llamar la atención sobre el dato a introducir.

Y por último la sentencia «PAUSE 200» temporiza, aproximadamente, la ejecución del programa durante cuatro segundos.

La estructura del programa es la siguiente.

- 10 : Comentario con el nombre del programa.
- 20 : Asignación de los colores de la pantalla, azul, para el borde y papel, y amarillo para los caracteres.
- 50 — 110 : Dibujo del contorno con los gráficos predefinidos.
- 130 — 210 : Visualización de los campos de la ficha.
- 220 : Temporización.
- 240 — 500 : Entrada de los datos y visualización de estos en los campos correspondientes.
- 510 : Utilización del canal de comunicación para visualizar el informe «Fin de edición».
- 520 : Temporización.

COMANDOS DE CONTROL

RJN

Acceso al tedado

INT



MOD0 **K** VERIFY

Definición

«RUN» se utiliza normalmente como comando directo y permite al usuario, mediante su ejemplo, ejecutar un programa editado en lenguaje BASIC.

La estructura general de este comando es:

SENTENCIA	ARGUMENTO
RJN	Nº de línea

Ejemplos:

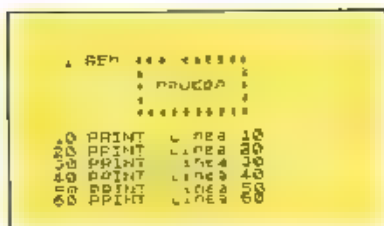
- RUN 30
- RUN 10
- RUN

Si no se especifica número de línea, e intérprete BASIC tomará, por defecto, como primera línea a ejecutar, la de numeración más baja.

Quando el número de línea especificado en el argumento no exista, la ejecución del programa comenzará en la línea siguiente; si esta tampoco existiera, por que se encuentra fuera de la zona de nuestro programa BASIC, no se ejecutará el programa y además aparecerá el mensaje:

0 0 0 1

Edite el siguiente programa:



Ejecútelo de las siguientes formas y compare los resultados:

- | | | |
|---|-----|----|
| 1 | RUN | |
| 2 | RUN | 10 |
| 3 | RUN | 20 |
| 4 | RUN | 30 |

100

- RUN 70

compare también los resultados proporcionados por los siguientes comandos directos:

- RUN 35
- RUN 40

Si en el argumento se especifica un número de línea comprendido entre "32768" y "61439" aparece el mensaje de error:

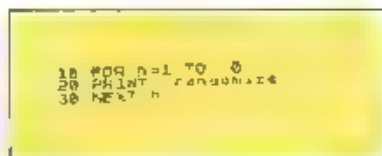
N Statement lost @ 255

si es mayor a este último valor (61439) aparece:

Integer out of range, 0 1

Una de las particularidades de la sentencia «RUN» es que borra la pñtal a antes de ejecutar el programa almacenado en memoria.

Ejecute varias veces segun
 las el siguiente programa y
 observe el efecto



Otra de las particularidades, es que borra todas las varia-

bles que hasta ese momento
estuvieran de fin das.

Employer

- Edite estas dos líneas:

```
10 PRINT a
20 PRINT b
```

Agisne unos valores a las variables «a» y «b» con comandos directos, por ejemplo,

LET a - 20
LET b - 127

- Compruebe los contenidos de dichas variables con:

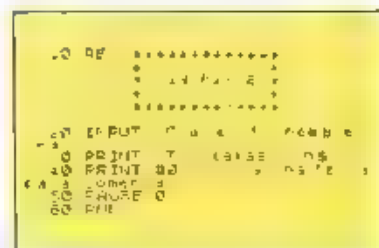
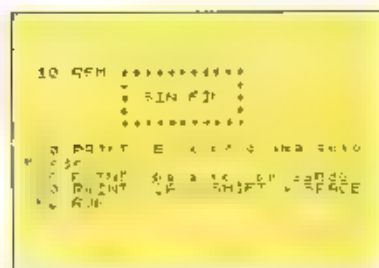
```
PRINT a
PRINT b
```

- Ejecute el programa con «RUN», observará que en esta ocasión aparece el mensaje:

```
2 variable not found 10 1
```

ya que al ejecutarse «RUN» se han borrado las variables «a» y «b».

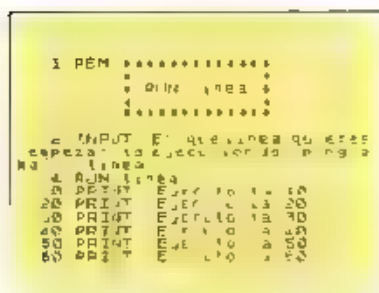
La sentencia «RUN» también puede ser usada como línea dentro de un programa.



en estos dos ejemplos, el programa comienza a ejecutarse de nuevo al llegar a la última sentencia, de esta forma se crea un bucle sin fin.

El argumento también puede ser una variable numérica previamente definida.

Ejemplo.



Una vez ejecutado el programa, éste pide que le introduzcamos el n.º de línea de la nueva ejecución, valor asignado a la variable «linea», la sentencia «RUN linea» lo hace a partir de este valor.

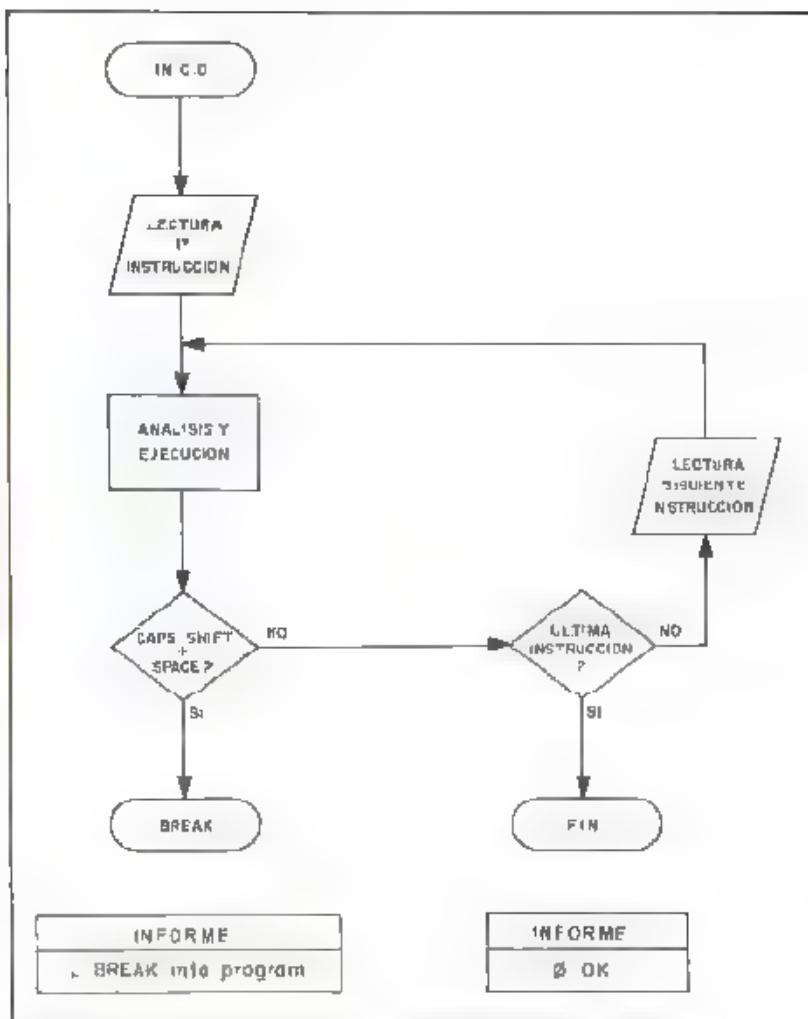
BREAK

Acceso al teclado



Definición

La función «BREAK» provoca una ruptura en la ejecución de un programa, en el acceso a los periféricos «impresora» y «cassette» y en los listados de más de 22 líneas, es decir, en



Análisis sentencia «Break».

aquellos en que aparece el mensaje:

scrib?

Esta ruptura sólo provoca una *interrupción* en la ejecución del programa, es decir que *no borra* el contenido de la memoria. En la mayoría de los casos se podrá continuar con ella, utilizando el comando «CONTINUE» (CONT).

ADVERTENCIA

Si se está ejecutando una sentencia del tipo «INPUT», no se puede provocar la ruptura del programa, esto se consigue utilizando otra técnica que posteriormente será descrita.

El intérprete BASIC al *terminar* de ejecutar una instrucción verifica si están pulsadas las teclas «CAPS SHIFT» y «SPACE», si no lo están continúa con la ejecución de la siguiente instrucción, y si por el contrario, lo están, provoca su interrupción.

En aquellas instrucciones en que el tiempo de ejecución es prolongado, es necesario *mantener* estas teclas oprimidas hasta que aparezca el informe correspondiente.

Ejemplo

```

10 BEEP 2 0
20 BEEP 3 20
30 BEEP 1 5
40 BEEP 3 5
  
```

Ejecute estas sentencias y utilice la función «BREAK»

Dependiendo de la situación en que se utilice «BREAK», existen dos tipos de informes. Cuando se utiliza para interrumpir un programa, el informe visualizado en pantalla es

└ BREAK into program

En los restantes casos, con sólo mantener pulsada la tecla «SPACE» (BREAK) se consigue la interrupción, y el mensaje presentado por el ordenador es.

└ BREAK — CONT repeats

La diferencia entre estos dos mensajes será explicada con detalle al tratar la sentencia «CONTINUE» (CONT).

Al final del informe aparece la línea y el número de sentencia, dentro de la línea donde se produjo la interrupción

STOP

Acceso al teclado

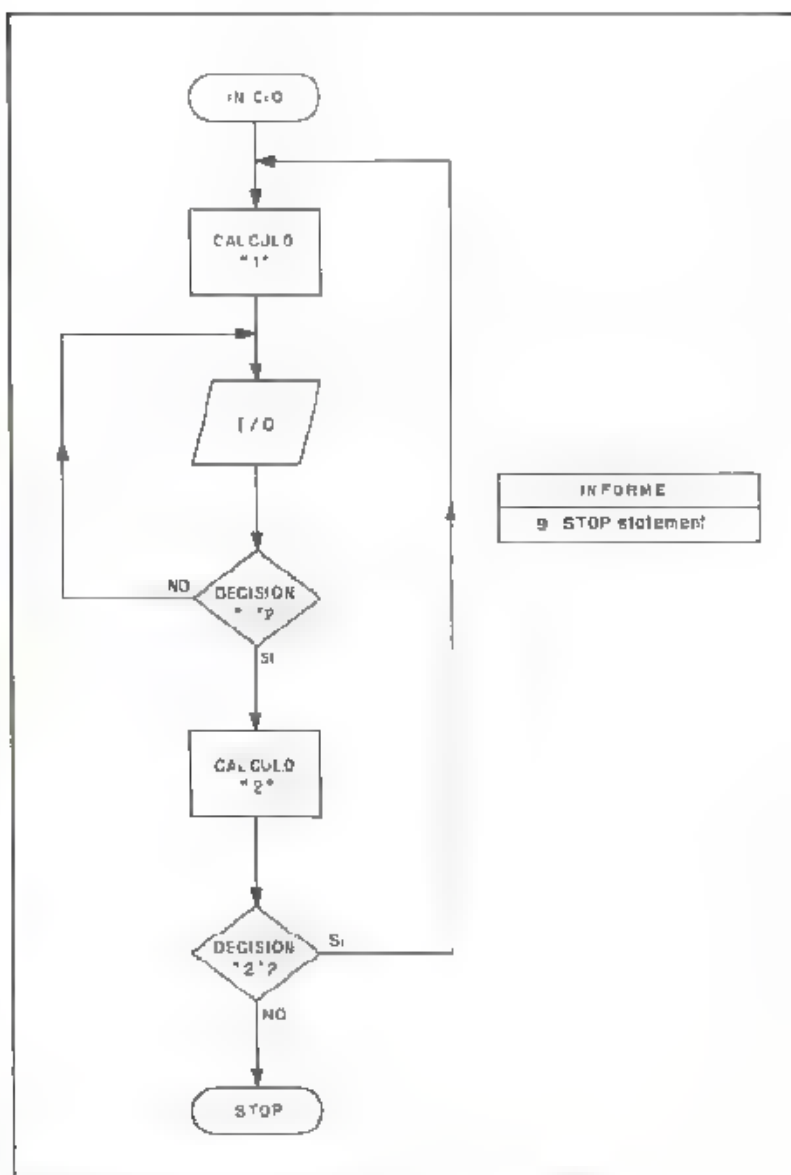


CONT



Tipo de sentencia

Comando de programación.



Ejemplo sentencia «Stop».

Definición

A pesar de ser un comando de programación, «STOP» es tratado en este capítulo, ya que su función es de control, cuando se ejecuta se interrumpe el programa.

«STOP» no tiene sentido como comando directo, por lo que debe ser editado como línea de programa *sin argumento*.

Ejemplo:

4000

STOP

cuando el Intérprete BASIC analiza esta sentencia, se detiene en la línea 4000 y presenta el mensaje

9 STOP statement 4000 1

Esta sentencia puede ser utilizada en diversas ocasiones, pero principalmente:

a) Para separar diversas rutinas independientes dentro de un solo programa.

Ejemplo:

Edite el programa «1» que simula una calculadora básica.

Las sentencias «STOP» separan las rutinas de «suma», «resta», «multiplicación» y «división». Para acceder a las diversas rutinas se utiliza la sentencia «RUN» y como argumento la variable «código», el valor de esta se asigna con el «INPUT» de la línea «8».

- b) Para separar un programa principal de las subrutinas. Estas se estudiarán en otro capítulo.
- c) Cuando se desea que se interrumpa la ejecución de un programa en función del resultado de una comparación. Se utiliza conjuntamente con el par de sentencias «IFTHEN».

Ejemplo:

```

10 REM *****
   : COMPARR :
   : ***** :
20 INPUT "Numero A" A1
30 PRINT A1
40 INPUT "Numero B" A2
50 PRINT A2
60 IF A1=A2 THEN STOP
70 GO TO 10

```

se producirá la interrupción del programa, cuando las variables «a1» y «a2» sean iguales.

- d) En técnicas especiales de depuración de programas.
- e) También se utiliza para provocar la ruptura de un programa, en una sentencia del tipo «INPUT».

Ruptura del «INPUT»

Para interrumpir la ejecución de un programa en una sentencia «INPUT», es necesario utilizar una serie de técnicas, dependiendo estas del tipo de «INPUT».

- INPUT numérico.
- INPUT de cadena.
- INPUT LINE.

PROGRAMA 1

```

10 REM *****
   : CURSD BASIC :
   : ***** :
   : CALCULADORA :
   : ***** :
20 BORDER 4: PAPER 4 INK 1: C
LS
30 REM *****
   : OPCIONES :
   : ***** :
40 PRINT AT 3,10;"CALCULADORA"
50 PRINT AT 7,4;"CÓDIGO OPE
RACION"
60 PRINT AT 8,4;"_____
"
70 PRINT AT 10,6;"14 SUMA
"
80 PRINT AT 12,6;"22 REST
A"
90 PRINT AT 14,6;"30 MULT
PLICACION"
100 PRINT AT 16,6;"38 DIVI
SION"
110 INPUT "Introduzca código de
operación ";codigo
120 CLS
125 RUN codigo*10
130 REM *****
   : SUMA :
   : ***** :
140 PRINT AT 3,13;"SUMA"
150 INPUT "Sumando 1? ";suma1
160 INPUT "Sumando 2? ";suma2
170 LET suma=suma1+suma2
180 CLS
190 PRINT suma1;" + ";suma2;" =
" suma
200 STOP
210 REM *****
   : RESTA :
   : ***** :
220 PRINT AT 3,12;"RESTA"
230 INPUT "Minuendo? ";min
240 INPUT "Sustraendo? ";sus
250 LET resta=min-sus
260 CLS
270 PRINT min;" - ";sus;" = ";r
esta

```



```

290 STOP
290 REM
      * ***** *
      *      *
      * MULTIPLICACION *
      *      *
      * ***** *

300 PRINT AT 3,8;"MULTIPLICACION"
310 INPUT "Multiplicando? ";mul
320 INPUT "Multiplicador? ";mul
330 LET multiplicacion=mul*mul
340 CLS
350 PRINT mul;" * ";mul;" = ";
multiplicacion
360 STOP
370 REM
      * ***** *
      *      *
      * DIVISION *
      *      *
      * ***** *

380 PRINT AT 3,12;"DIVISION"
390 INPUT "Dividendo? ";div
400 INPUT "Divisor? ";div1
410 LET division=div/div1
420 CLS
430 PRINT div;" / ";div1;" = ";
division
440 STOP

```

PROGRAMA 2

```

10 REM *****
   *      *
   * CURSO BASIC *
   *      *
   * ***** *
   * !!! NEW !!! *
   *      *
   * ***** *

20 BORDER 6: PAPER 6: INK 1 C
LS
30 REM *****
   *      *
   * DIBUJO CRONO *
   *      *
   * ***** *

40 PRINT AT 10,11; PAPER 2; IN
K 4; FLASH 1; "
50 FOR n=11 TO 13
60 PRINT AT n,11; PAPER 2; INK
4; FLASH 1; " "; AT n,19; "
70 NEXT n
80 PRINT AT 14,11; PAPER 2; IN
K 4; FLASH 1; "
90 PRINT AT 12,13; "10:00"

```

Cuando son del tipo numérico basta simplemente con teclear la sentencia «STOP» (SYMBOL SHIFT + A) y «ENTER», inmediatamente se provoca la ruptura del programa y aparece el mensaje:

1 STOP n INPUT

Ejemplo:

```

10 INPUT "a";a
20 PRINT a
30 GO TO 10

```

Si intenta utilizar la función «BREAK», observará que no sirve en estos casos.

Puede obtener la ruptura también, de una forma menos elegante, tecleando letras aleatoriamente, de esta manera el intérprete BASIC al analizar la entrada de datos y comprobar que no corresponde con un valor numérico o con una variable previamente definida, visualizará el mensaje:

2 Variable not found

Cuando el «INPUT» es del tipo alfanumérico, la técnica es ligeramente distinta.

Ejemplo:

```

10 INPUT "a";a$
20 PRINT a$
30 GO TO 10

```

Intente introducir «STOP», observará que el programa no se interrumpe, ya que la variable alfanumérica «a\$» asume el código correspondiente al token «STOP», y se ejecuta la siguiente instrucción, y así sucesivamente. La única manera de introducir «STOP», sin que lo tome como valor alfanumérico, es eliminando las comillas ("").

Para borrar las comillas, existen dos métodos. El primero es utilizando la función «DELETE», de esta forma desaparece la *comilla* situada a la izquierda del cursor. A partir de este instante ya se puede introducir la sentencia «STOP»; el mensaje que aparece es también:

H STOP in INPUT

En el segundo método se utiliza la función «EDIT», de esta manera desaparecen ambas comillas y al igual que en el caso anterior, se puede introducir «STOP».

Para cortar un «INPUT LINE», la filosofía es totalmente distinta, ya que ni se permite introducir «STOP», ni eliminar las comillas, simplemente por que estas no existen.

Ejemplo

```
10 INPUT " "; LINE a$
20 PRINT a$
30 GO TO 10
```

La única manera de salir del programa anterior, una vez ejecutado, es utilizando el cursor de desplazamiento inferior («CAPS SHIFT» + «6»). El mensaje presentado, al igual que en las anteriores situaciones es

H STOP in INPUT

CONTINUE

Acceso al teclado

L PRINT



MODOS K

PAPER

94 MICROBASIC

```
100 PRINT AT 3,0,"Este programa
se autodestruirán" al llegar
a la cuenta a 00 00"
110 REM
```

```
*****
*
* CUENTA ATRAS *
*
*****
```

```
120 FOR M=9 TO 0 STEP -1
130 PRINT AT 12,13;"0";M
140 FOR S=59 TO 0 STEP -1
142 IF S<10 THEN PRINT AT 12,15
0;S;
GO TO 155
150 PRINT AT 12,15;S
155 BEEP 0.02,20
160 NEXT S
170 NEXT M
180 REM
```

```
*****
*
* !!! BOOM !!! *
*
* X X X 1 2 3 4 5 6 7 8 9 0 *
*****
```

```
190 FOR n=0 TO 7
200 BORDER n
210 BEEP 0.03,n
220 NEXT n
230 CLS
235 FOR n=1 TO 50
240 PRINT OVER 1,AT 10,8,"!!!
BOOM
245 NE= n
250 REM
```

```
*****
*
* ' NEL *
*
*****
```

260 NEL

Definición

El comando directo «CONTINUE» se reconoce en el teclado por su forma abreviada «CONT». La utilidad de este comando es continuar con la ejecución de un programa que, debido a un informe de error o a un «BREAK» se ha interrumpido. Este comando no precisa argumento.

Cuando la interrupción se ha debido a un «BREAK» con informe:

L BREAK into program

o a una sentencia «STOP».

9 STOP statement


el comando «CONTINUE» comienza a ejecutar el programa a partir de la siguiente instrucción, en que se produjo la ruptura.

Sin embargo, cuando el informe presentado por «BREAK» es:

D BREAK — CONT repeats

o se visualiza

H STOP in INPUT

TIPO DE " INPUT "	METODO	
NUMERICO	STOP	
ALFANUMERICO	ELIMINAR	DELETE
	COMILLAS	EDIT
	+ STOP	
LINE		CAPS SHIFT + 6

Ruptura en «Input».

«CONTINUE» repite la ejecución en la misma línea donde se provocó la interrupción.

Ejemplo:

Introduzca las siguientes líneas

```
10 LOAD
20 PRINT cargado
```

cuando está en la fase de «carga» pulse la tecla «SPACE». Una vez interrumpido teclee «CONT», el programa volverá a ejecutar la sentencia «LOAD».

Si el programa se ha interrumpido debido a un error, podemos subsanar el problema momentáneamente y continuar con la ejecución.

Ejemplo:

```
0 REM *****
  FALLO
*****
20 INPUT "Coordenada X" X
30 PRINT "Z"
40 INPUT "Coordenada Y" Y
50 PRINT "Z"
60 GO TO 10
```

Este programa una vez ejecutado e introducido el valor de la «coordenada X» presenta un fallo en la línea 30, ya que no está definida la variable «Z». Defínala con un comando directo como por ejemplo:

```
[ LET Z = 5 ]
```

y tecleando «CONTINUE» volverá a ejecutarse el programa a partir de la línea 30

«CONTINUE» no se puede emplear con comandos directos. Se pueden distinguir tres casos.

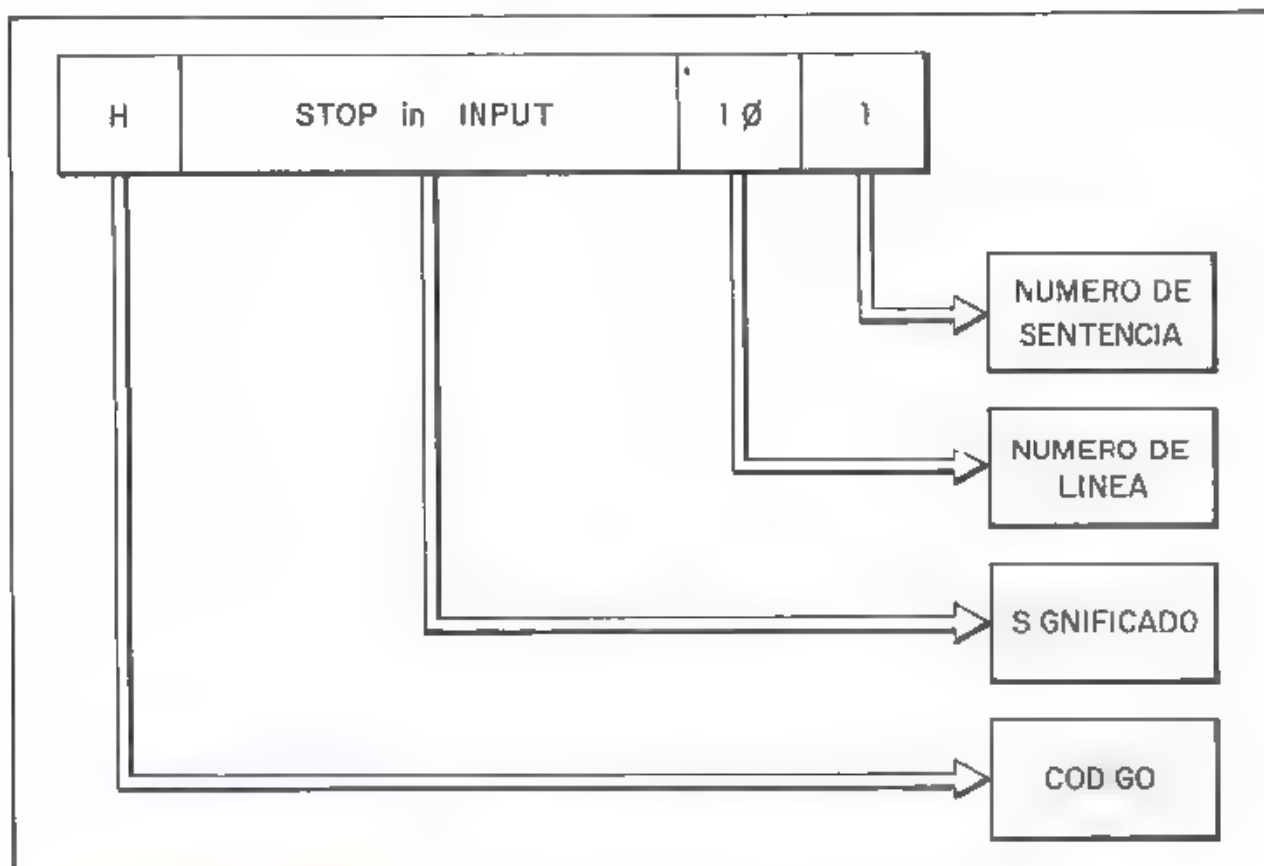
a) Cuando se interrumpe en la primera sentencia

Ejemplo.

```
LOAD " LET a = 2 PRINT a
```

si se pulsa «SPACE» (BREAK) y se pretende continuar con la ejecución de las restantes sentencias, el programa pierde el control y se queda un bucle sin fin, para salir de él pulse la tecla «CAPS SHIFT» + «SPACE».

b) Cuando se interrumpe en la segunda sentencia.



Informes.

Ejemplo.

```
PRINT "hola" LOAD " " PRINT "FIN"
```

En este caso al pretender continuar una vez realizada la interrupción se nos presenta el mensaje:

Ø OK

- c) Cuando se interrumpe en la tercera o siguiente sentencias.

Ejemplo:

```
INPUT > a PRINT a
LOAD " " LET b = 2
```

aparece el mensaje

N Statement lost

al intentar continuar con la ejecución una vez interrumpido en la sentencia «LOAD».

Informes de pantalla

Para comunicarnos el ordenador el resultado de la ejecución de los programas, tanto si han sido completados, interrumpidos, o con errores, utiliza estas dos líneas inferiores de la pantalla para enviarnos dichos informes.

Estos informes son visualizados de acuerdo a un formato.

CODIGO: Número comprendido entre «Ø» y «9» o letra de la «A» a la «B». Facilita la búsqueda en la tabla general de informes, apéndice B del manual (pág. 189).

SIGNIFICADO: Breve descripción del evento ocurrido. Para una mayor aclaración debe consultarse el manual.

LINEA: Número de línea donde se ha producido el evento.

SENTENCIA: Número de sentencia dentro de la línea.

Ejemplo:

Veamos el significado de informe:

```
H STOP in INPUT, 1 Ø 1
```

«H» es el código y significa que ha ocurrido una interrupción del programa en la sentencia primera de la línea 1 Ø, dicha interrupción ha sido debida a que se ha introducido «STOP» en una sentencia del tipo «INPUT».

Un informe con número de sentencia «2» se refiere a la sentencia situada a continuación del primer separador (:) o de la palabra clave «THEN», y así sucesivamente.

Los comandos directos al no poseer número de línea, se indican en los informes como línea Ø.

NEW

Acceso al tejado



MOD0 K

Definición

Generalmente este comando se utiliza de forma directa y no precisa de ningún argumento para poderse ejecutar.

«NEW» borra el programa o programas almacenados en memoria, también borra el valor de las variables definidas. Hay una serie de *variables de sistema* que no se ven afectadas por este comando, entre ellas los GDU o gráficos definidos por el usuario.

Debe utilizarse con mucho cuidado ya que de lo contrario, podríamos borrar un programa que aún no ha sido salvado, cosa no muy agradable por cierto.

Quando se ejecuta da la im-

presión de haber conectado el aparato de nuevo, ya que nos presenta el famoso mensaje inicial:

© 1982 Sinclair Research Ltd.

Puede incluirse con precaución dentro de un programa para dar por finalizada su ejecución y borrado.

El programa n.º «2» incorpora esta sentencia. "sálvelo antes de ejecutar"

C_S

Acceso al teclado



MODOK

Definition

El comando «CLS» puede ser utilizado tanto en modo di-

recto, como formando parte de un programa, no precisa de ningún argumento.

La función de este comando es borrar la pantalla de caracteres y gráficos, asumiendo ésta el color especificado en la última sentencia «PAPER» ejecutada con anterioridad. El color del *borde* de la pantalla no se ve afectado por esta sentencia.

Exemplos:

— Introduzca el siguiente programa:

```

10 REM *****
11
12
13 *****
14
15 *****
16
17 *****
18
19 *****
20
21 *****
22
23 *****
24
25 FOR I=1 TO 704
26 PRINT "I", I
27 NEXT I
28 REM *****
29
30 *****
31
32 *****
33
34 *****
35
36 *****
37
38 *****
39
40 *****
41
42 *****
43
44 *****
45
46 *****
47
48 *****
49
50 BORDER 2

```

este programa llena la zona de visualización con el símbolo «@», teclee el comando directo «GLS» y observe la pantalla.

- En el siguiente programa la sentencia «CLS» se utiliza para borrar la pantalla y asumir los colores «magenta» para el fondo y «amarillo» para los caracteres; el color del borde se asigna directamente con la sentencia «BORDER».

```

10 REM *****
      A ASIONAR COLADES 0
      *****
20 REM 10 11 12 13 14 15 16 17 18 19
30 REM 14 BOARD Verde 17
40 BOARD 4
50 REM 1 FONDO 199616 14
60 BOARD 1
70 REM 5 TINTA 1817, L 8 14
80 INK 5
90 CLS
99 PRINT AT 10 10, MICROHONEY

```



Ejemplo de listado.

LIST

Acceso al teclado



MODO K

Definición

«LIST» se utiliza normalmente como comando directo y permite obtener un listado del programa almacenado en memoria. La estructura general de esta sentencia es:

SENTENCIA	ARGUMENTO
LIST	Nº DE LÍNEA

Ejemplos:

- LIST 120
- LIST 30
- LIST

Cuando el argumento se omite, el intérprete BASIC ejecuta este comando a partir de la línea 1.

El listado del programa se visualiza en páginas de 22 líneas presentando en la parte inferior de la pantalla el mensaje:

scroll?

este mensaje, como ya recordará el lector de lo explicado anteriormente con la sentencia «PRINT», sirve para preguntarnos si queremos visualizar la siguiente página. Pulsando las teclas «N», «SPACE» o «STOP» («SYMBOL SHIFT» + «a») el listado se interrumpirá

y se nos presentará el mensaje

D BREAK CONT repeats

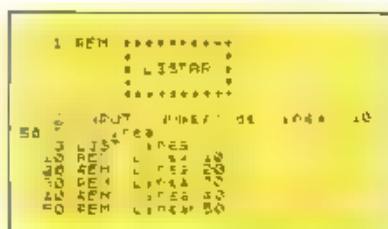
pulsando cualquier otra tecla se visualizará la siguiente página y así sucesivamente hasta que se termine el listado (mensaje 0 OK).

Cuando como argumento se introduce, por error, un número decimal, el intérprete BASIC redondea este valor hasta el número entero más próximo: si tiene un programa almacenado en la memoria, ejecute estos dos comandos directos y observe los resultados.

LIST 10.2
LIST 10.5

Si se especifica un número de línea inexistente, el comando «LIST» empezará a ejecutarse a partir de la siguiente.

El argumento también puede ser una variable numérica previamente definida. En el siguiente ejemplo, la instrucción «LIST línea» visualiza el listado a partir del valor asignado a la variable «línea».



LIST y EDIT

En el capítulo «2» dedicado a la edición de programas y

corrección de los posibles errores, se estudió un método para corregir líneas de programa una vez editadas. Este método consistía en desplazar con los cursores (y) el prompt «>» hasta situarlo en la línea que queríamos corregir; pero ¿qué pasa si tenemos el cursor en la línea 4000 y queremos corregir la 200? como vemos este método no es efectivo ya que perderíamos mucho tiempo desplazando cursores; en estos casos resulta más interesante utilizar el comando «LIST».

Para situar el prompt «>» en la línea que deseamos corregir basta simplemente con pedir un listado a partir de dicha línea, por ejemplo si deseamos corregir la línea 200, introduciremos el comando directo.

LIST 200

lógicamente si el listado es largo aparecerá el mensaje

scroll?

pulsando la tecla «N», «SPACE» o «STOP» el listado se interrumpirá. El prompt ya lo tenemos situado en la línea 200; para corregirla utilizaremos la función «EDIT», con lo que nos pasará a la parte inferior de la pantalla. A partir de este momento podremos corregirla utilizando los cursores (y) y la función «DELETE». Una vez terminada la modificación, pulsando «ENTER» volverá a la parte superior. ■

SALTOS INCONDICIONALES Y CONDICIONALES

Hay ocasiones en que por razones de estructura de un programa, interesa que las instrucciones no se ejecuten de forma secuencial, es decir, una detrás de otra, sino que, por el contrario realicen *saltos*; éstos pueden clasificarse dependiendo de su función en:

- INCONDICIONALES.
- CONDICIONALES.

Como su propio nombre indica, un salto *incondicional* es aquel que salta directamente al número de línea especificado en el argumento, sin embargo, los condicionales necesitan que se cumpla previamente la condición prevista en la instrucción.

GO TO

Acceso al teclado



MODO **K**

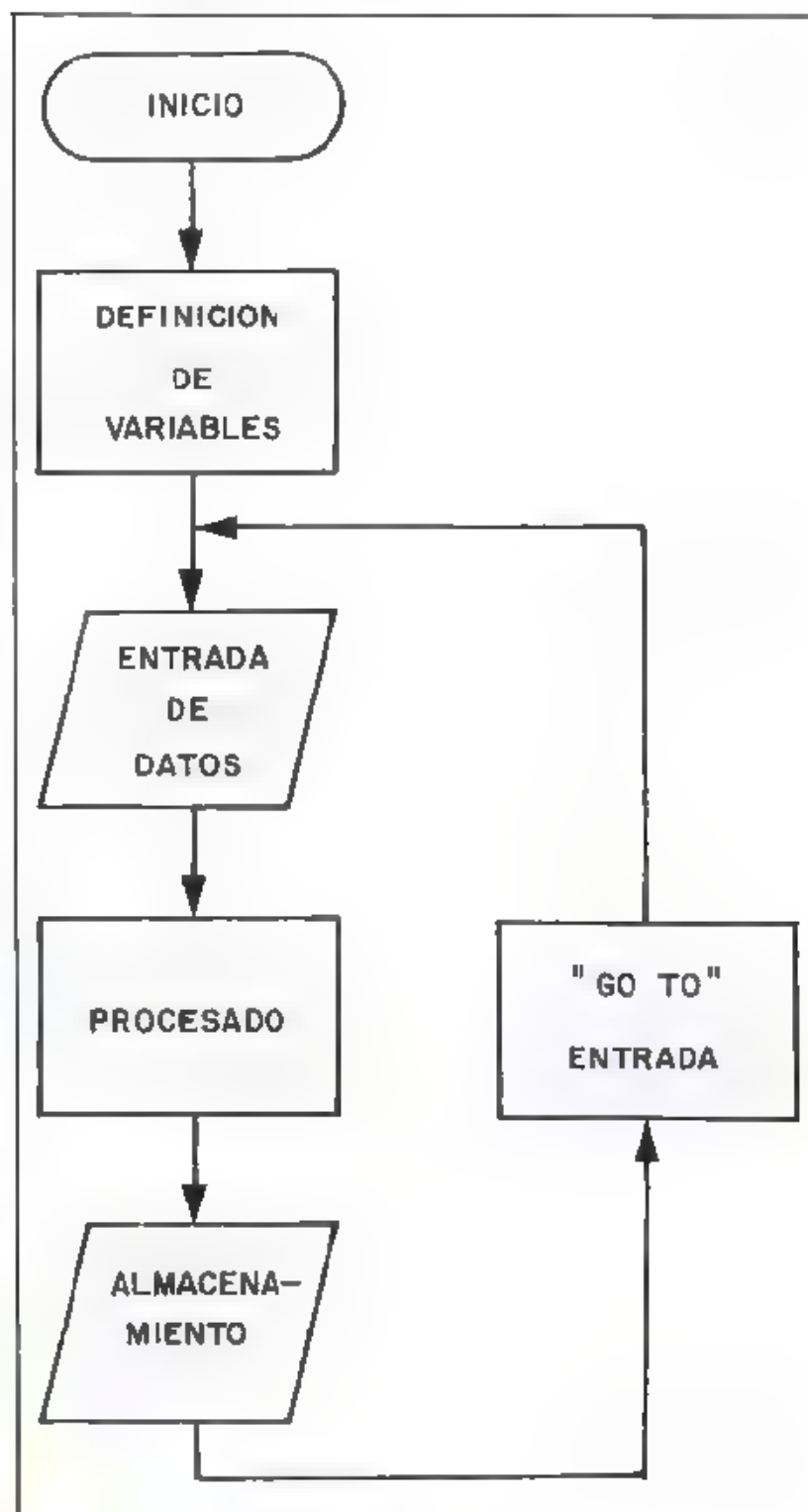
Tipo de sentencia

Comando de programación

Definición

La sentencia «GO TO» realiza los saltos incondicionales dentro de un programa, su estructura general es:

SENTENCIA	ARGUMENTO
GO TO	Nº de línea



Ejemplo sentencia "GO TO".

Ejemplos:

- GO TO 30
- GO TO 70

Una sentencia de este tipo transfiere la ejecución de programa a la línea especificada en su argumento.

En el siguiente programa, al analizar el intérprete BASIC la instrucción «40», la siguiente que ejecutará será la «10»

```

10 REM *****
      :      GO TO      :
      : *****        :
20 INPUT "Escriba lo que quiera"
30 PRINT "ENTER"
40 GO TO 10
  
```

En este otro se ha incluido un índice asignado a la variable «potencia» que se incrementa en uno cada vez que se ejecuta la línea «50», este índice sirve tanto para indicar el número de veces menos uno, que se realiza la operación matemática (2^n) como para utilizarse como potencia de la misma.

```

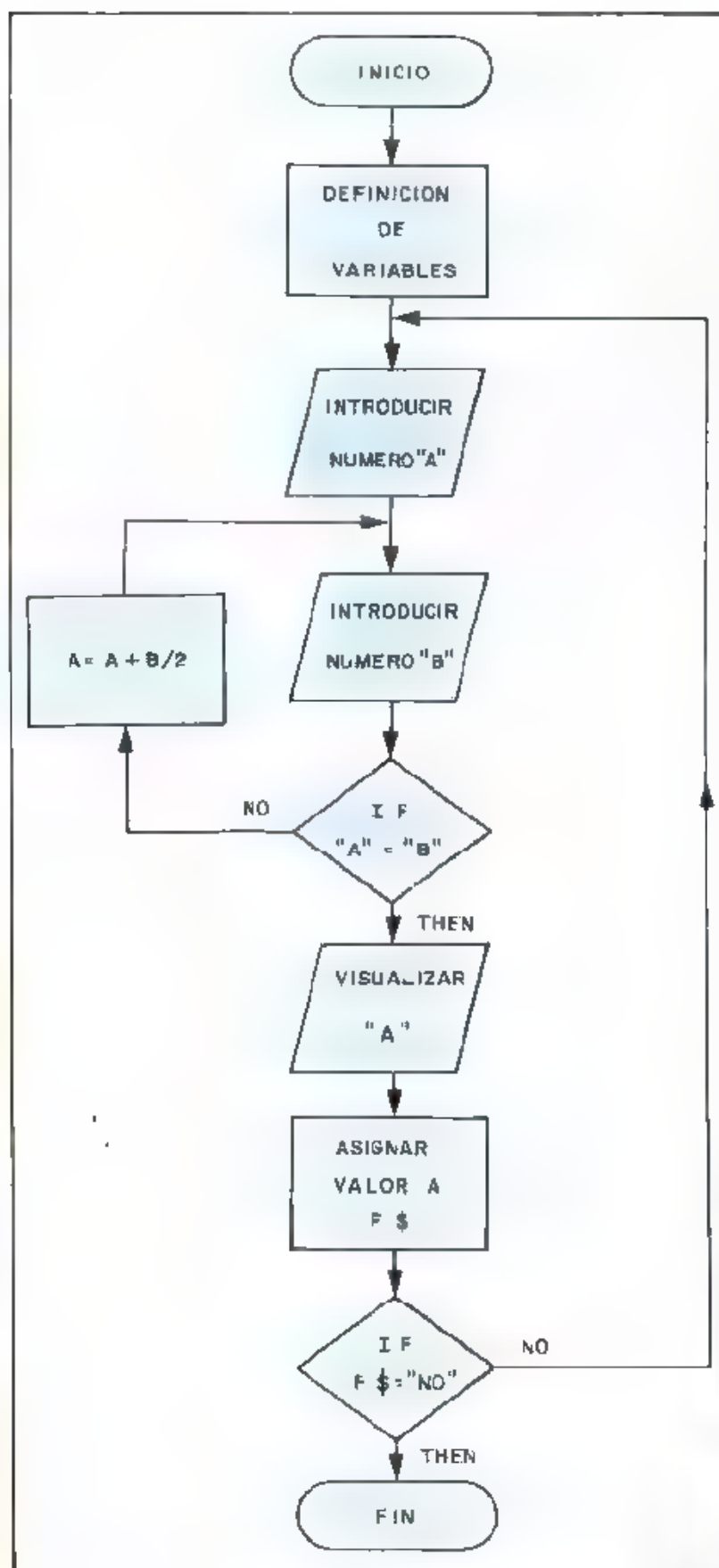
10 REM *****
      :      POTENCIA  :
      : *****        :
20 LET potencia = 0
30 PRINT potencia
40 PRINT "2^potencia"
50 LET potencia=potencia+1
60 GO TO 30
  
```

Debe poner atención al calcular el número de línea donde desea que se realice el salto, ya que podrían no ejecutarse ciertas líneas intermedias.

Ejemplo:

```

10 REM *****
      :      INCORRECTO :
      : *****        :
20 LET a$= "MICRO"
30 LET b$= "HOBBI"
40 LET c$= "AFORMOSAL"
50 PRINT a$
60 PRINT b$+ " "
70 PRINT c$
80 GO TO 50
  
```



Ejemplo salto y ejecución condicional (IF . THEN...).

en este ejemplo, solamente la primera vez se ejecuta el programa correctamente; en las siguientes, la variable «a\$» no se visualiza, por tanto la línea 00 debería ser:

GO TO 50

«GO TO» también puede ser utilizado como comando directo, esta aplicación es bastante interesante, ya que permite ejecutar un programa sin alterar el contenido de las variables hasta ese momento definidas. En la depuración de programas se utiliza frecuentemente en sustitución de comando «RUN» que lo borra todo.

El argumento puede ser una variable de tipo numérico

```

1 REM *****
2 JRR ZMBLE
*****
4 ZMP 10 20 30 40 50 0 60
5 GO TO 10
6 PRINT "A QUE VALOR HAGO?"
7 INPUT "B" B
8 IF B=1 THEN GO TO 20
9 IF B=2 THEN GO TO 30
10 IF B=3 THEN GO TO 40
11 IF B=4 THEN GO TO 50
12 IF B=5 THEN GO TO 60
13 PRINT "FIN"
14 GO TO 4

```

una vez introducida la variable «línea» el control de ejecución salta hasta el valor especificado en ella.

IF ... THEN ...

Acceso al teclado



Tipo de sentencia

Comando de programación

Definición

El grupo de sentencias «IF» y «THEN» permiten realizar los saltos o ejecutar una serie de instrucciones de una manera *condicional*, es decir, en función del resultado de una comparación.

Las estructuras básicas son

a) Salto condicional.

SENTENCIA	ARGUMENTO
IF condición	THEN GO TO...

Ejemplo.

```

10 REM *****
11 SALTO
*****
20 INPUT "NUMERO" B
30 IF B=1 THEN GO TO 70
40 LET A=B/2
50 GO TO 60
60 PRINT "A" A
70 PRINT "FIN"

```

cuando las variables «a» y «b» sean iguales, la ejecución del programa continuará en la línea 70, si no, se asignará a la variable «a» el valor de «a + b/2» y posteriormente, ejecuta un salto incondicional a la línea 30.

b) Ejecución condicional.

SENTENCIA	ARGUMENTO
IF condición	THEN instrucciones..

Ejemplo.

```

10 REM *****
11 EJECUCION
*****
20 PRINT "MICROBASIC"
30 IF B=1 THEN PRINT "A"
40 IF B=2 THEN PRINT "B"
50 IF B=3 THEN PRINT "C"
60 GO TO 20

```

si no se cumple que la variable de cadena «f\$» sea igual a «NO» se ejecuta la instrucción siguiente, si por el contrario lo son, se ejecutan las sentencias que acompañan al «THEN».

De lo explicado hasta este momento, se desprende que la sentencia «IF ... THEN ...» es como una encrucijada con dos caminos, donde el ordenador tiene que elegir uno de ellos.

OBSERVACION

Si se utiliza la sentencia «GO TO», dentro de la lista de instrucciones que deben ejecutarse si se cumple la condición impuesta en el «IF», ésta deberá ser colocada la última, ya que de lo contrario, quedarían sin ejecutar algunas instrucciones.

Ejemplo.

```

10 F mes B THEN PRINT
"AGUSTO" GO TO 120
LET mes 0

```

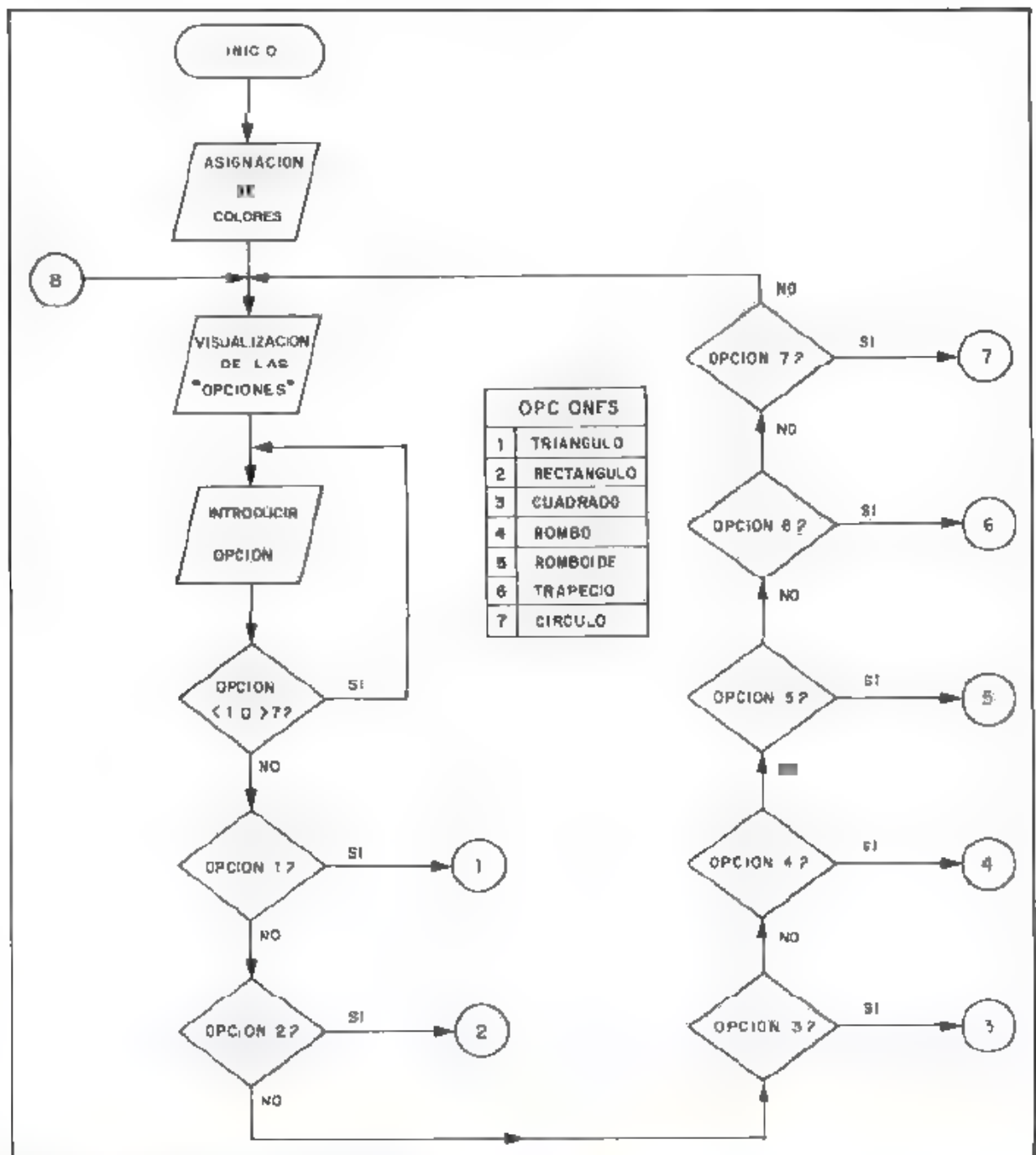
la asignación del valor «0» a la variable «mes» no se realiza, ya que antes se ejecuta una instrucción de salto incondicional a la línea 120.

Para realizar las comparaciones, puede utilizarse cualquiera de los operadores *relacionales*:

>	MAYOR QUE ...
<	MENOR QUE ..
> =	MAYOR O IGUAL
< =	MENOR O IGUAL
< >	DISTINTO

Ejemplos.

- IF a > b THEN ...
- IF J\$ < S\$ THEN ...
- IF n > - K + 1 THEN
- IF K0 < = 7 THEN ..
- IF P3 < > "SI" THEN...



Programa "Áreas" menú de opciones.

PROGRAMA 1

```

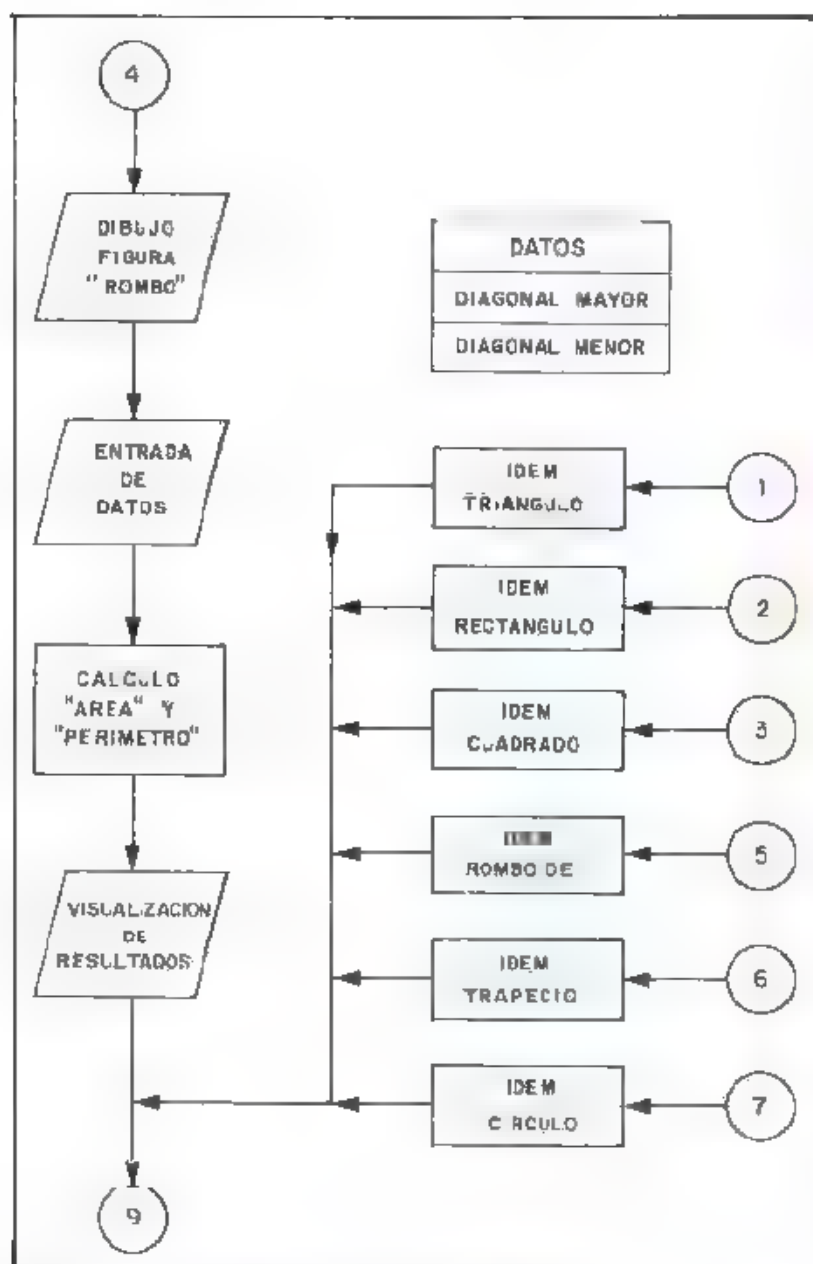
10 REM *****
   * AGENDA *
   * *****
20 PRINT AT 0,4, "NOMBRE"
25 PRINT "TELEFONO"
27 LET indice=1
30 INPUT "Nombre >>> ", LINE n

```

```

40 IF n$="FIN" OR n$="fin" THEN
   N GO TO 1000
50 PRINT n$,
50 INPUT "Telefono >>> ", telef
60
70 PRINT telefono
80 IF indice=18 THEN GO TO 100
90 LET indice=indice+1
100 GO TO 30
1000 PRINT #0, AT 1,2, "(((( Fin
   de edicion >>> ))"
1010 PAUSE 0

```



Programa "Areas" desarrollo opción "Rombo".

Los operadores lógicos «AND» y «OR» son utilizados cuando hay una combinación de condiciones dentro de una sentencia «IF ... THEN ...». El operador «AND» implica que deben cumplirse todas y cada una de las condiciones.

Ejemplo:

```
10 IF n >= 5 AND r <= 9 THEN ...
```

solamente se cumplen las dos condiciones, cuando la variable

«n» tiene un valor comprendido entre «5» y «9».

Utilizando el operador «OR» basta solamente con que se cumpla una de las condiciones previstas en la comparación.

Ejemplo:

```
10 IF A$ = "A" OR B$ < > "C" OR n > 30 THEN ...
```

en este ejemplo, se cumple a condición general en cualquiera de los siguientes casos:

- Cuando la variable A\$ tenga el valor "A".
- Cuando el contenido de la variable B\$ sea distinto de "C".
- Cuando "n" sea superior a "30".
- O con cualquier combinación de los anteriores casos.

También pueden combinarse los operadores «AND» y «OR». Ejemplo:

```
10 IF (J$ = "NO" AND n < 10) OR (J$ = "SI" AND (p = 10 OR t = 7)) THEN ...
```

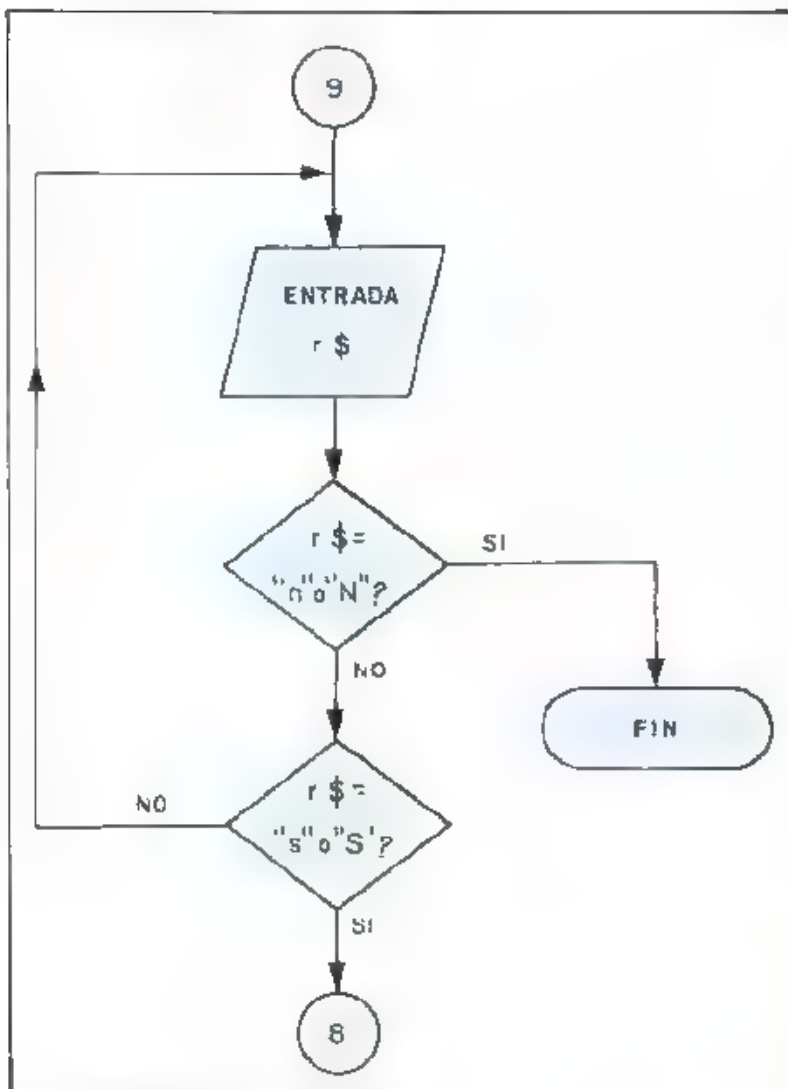
primero, se evalúan individualmente las condiciones encerradas entre paréntesis, y posteriormente, se evalúan entre sí los resultados parciales, por tanto, será necesario que se cumplan cualquiera de las siguientes condiciones:

- Cuando J\$ = "NO" y n < 10.
- Cuando J\$ = "SI" y p = 10.
- Cuando J\$ = "SI" y t = 7.
- Combinaciones de a, b y c.

El programa número «1» es un ejemplo de aplicación de las sentencias «IF ... THEN ...», acaba su ejecución cuando se teclean dieciocho nombres con sus correspondientes teléfonos o cuando se introduce la palabra «FIN» o «fin» en el instante que el ordenador espera un nombre.

Evaluación de las condiciones

Cuando una condición se cumple, es decir, que es verdadera, se le asigna el valor «1» (distinto de 0) y cuando es falsa, el valor 0 (igual a 0). Para evaluar una condición compleja, primero se evalúan una a una, asignando los valores «0» o «1» según corresponda; pos-



Programa "Areas" Rutina "Fin".

teriormente, por parejas hasta que sólo quede una condición, si el resultado es « \emptyset » el ordenador pasará a ejecutar la siguiente instrucción; si es «1», ejecutará antes las instrucciones contenidas en el «THEN».

Ejemplo:

IF (a > b OR c = 7) AND (l < > \emptyset AND p = a * 5) THEN

para los valores:

a = 4
b = 2
c = 6
l = 5 \emptyset
p = 2 \emptyset

Resolviendo por pasos:

a) (a > b) OR (c = 7)
(a > b) es verdadero (1), ya que «4» es mayor que «2».
(c = 7) es falso (\emptyset), ya que «6» no es igual a «7».

Realizando la operación lógica «OR» de los valores anteriores

1 OR \emptyset

observamos que el resultado es «1», ya que se cumple una de las condiciones. (Repasar el capítulo dedicado a «OPERADORES LÓGICOS»).

b) (l < > \emptyset) AND (p = a * 5)
(l < > \emptyset) es verdadero (1), ya que «5 \emptyset » es distinto de « \emptyset ».
(p = a * 5) es verdadero (1), por que «2 \emptyset » es igual a «4» por «5». Realizando la operación lógica «AND» de los valores anteriores «1» y «1»

1 AND 1

tenemos un resultado igual a «1» ya que se cumplen las dos condiciones.

c) (condición 1) AND (condición 2)

Como las dos condiciones se cumplen, el resultado global es también verdadero «1».

Asigne otros valores a las variables e intente resolver el resultado; si tiene algún problema o desea comparar los resultados ejecute el siguiente programa:

```

10 REM *****
11 VERDADERO FALSO
12 *****
20 INPUT A
30 INPUT B
40 INPUT C
50 INPUT T
60 INPUT P
70 IF (A > B OR C = 7) AND (T < >  $\emptyset$  AND P = A * 5) THEN PRINT "VERDA:"
80 PRINT "Otro"
90 GO TO 20
  
```

El valor de una condición también puede ser asignado a una variable, de la forma:

LET resultado = l\$ = "FIN"

la variable «resultado» tendrá el valor «1» cuando l\$ sea igual a la cadena «FIN», y « \emptyset » cuando l\$ tenga otro valor. Podríamos, por tanto, editar una sentencia del tipo:

IF resultado THEN ...

PROGRAMA 2

```

10 REM *****
*   CURSO BASICO   *
* *****          *
*   AREAS Y        *
*   PERIMETROS     *
* *****          *

20 BORDER 1: PAPER 1 INK 7: C
30 REM *****
*   OPCIONES       *
* *****          *

40 PRINT AT 1,0, "CALCULO DE AN
EAS @ MICROHOBBY:
50 PRINT AT 4,9, "MENU DE OPCIO
NES"
60 PRINT AT 5,7, "_____
70 PRINT AT 8,8: "1 - TRIANGULO
80 PRINT AT 10,8: "2 - RECTANGU
LO"
90 PRINT AT 12,8: "3 - CUADRADO
100 PRINT AT 14,8: "4 - ROMBO"
110 PRINT AT 16,8: "5 - ROMBATOPE
120 PRINT AT 18,8: "6 - TRAPECIO
130 PRINT AT 20,8: "7 - CIRCUNFE
RENCIA"
140 INPUT "Introduzca la opcion
deseada >>> ", option
150 REM *****
*   VERIFICACION   *
* *****          *

160 IF option=1 OR option>7 THE
N GO TO 140
170 CLS
180 IF option=1 THEN GO TO 250
190 IF option=2 THEN GO TO 450
200 IF option=3 THEN GO TO 700
210 IF option=4 THEN GO TO 970
220 IF option=5 THEN GO TO 1170
230 IF option=6 THEN GO TO 1380
240 IF option=7 THEN GO TO 1530
250 REM *****
*   TRIANGULO      *
* *****          *

260 PRINT AT 2,11, "TRIANGULO"
270 PRINT AT 3,11, "_____
280 PLOT INK 4,16,47
290 DRAW INK 4,63,0
300 DRAW INK 4,-32,73
310 DRAW INK 4,-31,-73
312 PRINT AT 17,5, "b"
314 PRINT AT 12,5, "h"
320 REM *****
*   ENTRADA DATOS  *
* *****          *

330 INPUT "Base >>> ", base
340 PRINT AT 7,11, "b", base
350 INPUT "Altura >>> ", altura
360 PRINT AT 9,11, "h", altura
370 REM *****
*   CALCULOS       *
* *****          *

```

```

380 LET area=base*altura/2
390 PRINT AT 12,11, "AREA
", area
400 LET lado=SQRT (altura^2+(bas
e/2)^2)
410 LET perimetro=lado*2+base
420 PRINT AT 14,11, "PERIMETRO
", perimetro
430 GO TO 9000
450 REM *****
*   RECTANGULO     *
* *****          *

460 PRINT AT 2,11, "RECTANGULO"
470 PRINT AT 3,11, "_____
480 PLOT INK 4,16,47
490 DRAW INK 4,63,0
500 DRAW INK 4,0,80
510 DRAW INK 4,-63,0
520 DRAW INK 4,0,-80
530 PRINT AT 17,5, "b"
540 PRINT AT 10,0, "h"
550 REM *****
*   ENTRADA DATOS  *
* *****          *

560 INPUT "Base >>> ", base
570 PRINT AT 7,10, "b", base
580 INPUT "Altura >>> ", altura
590 PRINT AT 9,13, "h", altura
600 REM *****
*   CALCULOS       *
* *****          *

610 LET area=base*altura
620 PRINT AT 12,13, "AREA
", area
630 LET perimetro=base*2+altura
640 PRINT AT 14,13, "PERIMETRO
", perimetro
650 GO TO 9000
700 REM *****
*   CUADRADO       *
* *****          *

710 PRINT AT 2,13, "CUADRADO"
720 PRINT AT 3,13, "_____
730 PLOT INK 4,16,47
740 DRAW INK 4,63,0
750 DRAW INK 4,0,63
760 DRAW INK 4,-63,0
770 DRAW INK 4,0,-63
780 PRINT AT 17,5, "b"
790 REM *****
*   ENTRADA DATOS  *
* *****          *

800 INPUT "Lado >>> ", lado
810 PRINT AT 7,13, "L", lado
910 REM *****
*   CALCULOS       *
* *****          *

920 LET area=lado^2
930 PRINT AT 12,13, "AREA
", area
940 LET perimetro=lado*4
950 PRINT AT 14,13, "PERIMETRO
", perimetro
960 GO TO 9000
970 REM

```

```

*****
* ROMBO *
*      *
*****

980 PRINT AT 2,14,"ROMBO"
990 PRINT AT 3,14,"-----"
1000 PLOT INK 4,40,40
1010 DRAW INK 4,24,40
1020 DRAW INK 4, 24,48
1030 DRAW INK 4,-24, 48
1040 DRAW INK 4,24,-48
1050 REM

*****
* ENTRADA DATOS *
*      *
*****

1060 INPUT "Diagonal mayor >>> " ,mayor
1070 PRINT AT 7,12,"D : ";mayor
1080 INPUT "Diagonal menor >>> " ,menor
1090 PRINT AT 9,12,"d - ";menor
1100 REM

*****
* CALCULOS *
*      *
*****

1110 LET area=mayor*menor/2
1120 PRINT AT 12,12,"AREA"
1130 LET lado=SOR ((mayor/2)2+ (menor/2)2)
1140 LET perimetro=lado*4
1150 PRINT AT 14,12,"PERIMETRO"
1160 GO TO 9000
1170 REM

*****
* ROMBOIDE *
*      *
*****

1180 PRINT AT 2,12,"ROMBOIDE"
1190 PRINT AT 3,12,"-----"
1200 PLOT INK 4,8,47
1210 DRAW INK 4,64,0
1220 DRAW INK 4,16,48
1230 DRAW INK 4,-64,0
1240 DRAW INK 4, 16, 48
1250 PRINT AT 17,5,"b"
1260 PRINT AT 19,4,"h"
1265 PRINT AT 13,0,"l"
1270 REM

*****
* ENTRADA DATOS *
*      *
*****

1280 INPUT "Base >>> " ,base
1290 PRINT AT 7,13,"b ";base
1300 INPUT "Altura >>> " ,altura
1310 PRINT AT 9,13,"h ";altura
1312 INPUT "Lado >>> " ,lado
1314 PRINT AT 11,13,"l ";lado
1320 REM

*****
* CALCULOS *
*      *
*****

1330 LET area=base*altura
1340 PRINT AT 14,13,"AREA"
1350 LET perimetro=lado*2+base*2
1360 PRINT AT 16,13,"PERIMETRO"
1370 GO TO 9000
1380 REM

*****
* TRAPECIO *
*      *
*****

```

```

1390 PRINT AT 2,12,"TRAPECIO"
1400 PRINT AT 3,12,"-----"
1410 PLOT INK 4,8,47
1420 DRAW INK 4,80,0
1430 DRAW INK 4,-16,64
1440 DRAW INK 4,-48,0
1450 DRAW INK 4,-16,-64
1460 PRINT AT 17,5,"B"
1470 PRINT AT 12,3,"D"
1480 PRINT AT 6,6,"b"
1490 REM

*****
* ENTRADA DATOS *
*      *
*****

1500 INPUT "Base mayor >>> " ,mayor
1510 PRINT AT 7,13,"B : ";mayor
1520 INPUT "Base menor >>> " ,menor
1530 PRINT AT 9,13,"b ";menor
1540 INPUT "Altura >>> " ,altura
1550 PRINT AT 11,13,"h ";altura
1560 REM

*****
* CALCULOS *
*      *
*****

1570 LET area=(mayor+menor)*altu
ra/2
1580 PRINT AT 14,13,"AREA"
1590 LET lado=SOR ((mayor-menor
/2)2+2+altura2)
1600 LET perimetro=mayor+menor+l
ado*2
1610 PRINT AT 16,13 "PERIMETRO"
1620 GO TO 9000
1630 REM

*****
* CIRCUNFERENCIA *
*      *
*****

1640 PRINT AT 2,9,"CIRCUNFERENCIA"
1650 PRINT AT 3,9,"-----"
1660 CIRCLE INK 4,40,88,32
1670 PRINT AT 9,3,"r"
1680 REM

*****
* ENTRADA DATOS *
*      *
*****

1690 INPUT "Radio >>> " ,radio
1700 PRINT AT 7,13,"r ";radio
1710 REM

*****
* CALCULOS *
*      *
*****

1720 LET area=PI*radio2
1730 PRINT AT 10,12,"AREA"
1740 LET perimetro=2*PI*radio
1750 PRINT AT 12,12,"PERIMETRO"
1760 GO TO 9000
1770 REM

*****
* MAS CALCULOS *
*      *
*****

9010 INPUT "Quiere continuar (S
o N) >>> " , LINE r$
9020 IF r$="n" OR r$="N" THEN ST
OP
9030 IF r$="s" OR r$="S" THEN CL
S
9040 GO TO 9010

```

Programa

Como programa de repaso de las sentencias «GO TO» e «F... THEN...», se propone el programa número «2». Sálvelo, por ejemplo, de la forma:

SAVE "AREAS" LINE 10

Este programa calcula el área y el perímetro de las siguientes figuras geométricas.

- TRIANGULO.
- RECTANGULO.
- CUADRADO.
- ROMBO.
- ROMBOIDE.
- TRAPPECIO.
- CIRCULO.

El programa, al autoejecutarse, presenta en pantalla un menú con las diversas opciones, seleccionando una de ellas, pasaremos a una pantalla en la que aparecerá dibujada la figura geométrica correspondiente, ésta es realizada con ayuda de

las sentencias «PLOT», «DRAW» o «CIRCLE». Una vez introducidos los datos (lado, altura,...) que nos pide el ordenador, los resultados correspondientes a «área» y al «perímetro» serán visualizados.

Para retomar al menú principal debe pulsar «S» o «s», pulsando «n» o «N» el programa se detendrá presentando el mensaje:

9 STOP statement 90 20.2

La estructura del programa es.

10	Comentario con el nombre del programa
20	Asignación de los colores de la pantalla.
30-130	: Menú de opciones.
140	: Entrada de «opción»
150-240	: Verificación y selección.
250-314	: Dibujo de <i>triángulo</i> .
320-360	: Entrada de datos, (base y altura)
370-430	: Cálculo y visualización

440	: Salto de la rutina «F No».
450-540	: Dibujo del <i>rectángulo</i> .
550-590	: Entrada de datos, (base y altura)
600-640	: Cálculo y visualización.
650	: Salto a la rutina «FIN».
700-780	: Dibujo del <i>cuadrado</i> .
790-900	: Entrada de datos, (lado)
910-950	: Cálculo y visualización.
960	: Salto a la rutina «FIN».
970-1040	: Dibujo del <i>rombo</i> .
1050-1090	: Entrada de datos, (diagonal mayor y diagonal menor)
1100-1150	: Cálculo y visualización.
1160	: Salto a la rutina «FIN».
1170-1265	: Dibujo del <i>romboide</i> .
1270-1314	: Entrada de datos, (base, altura y lado).
1320-1360	: Cálculo y visualización.
1370	: Salto a la rutina «FIN».
1380-1480	: Dibujo del <i>trapezio</i> .
1490-1550	: Entrada de datos, (base mayor, base menor y altura).
1560-1610	: Cálculo y visualización.
1620	: Salto a la rutina «FIN».
1630-1670	: Dibujo del <i>círculo</i> .
1680-1700	: Entrada de datos, (radio).
1710-1750	: Cálculo y visualización
1760	: Salto a la rutina «FIN».
9000-9040	: Rutina «FIN».

BUCLAS

Al analizar un programa, sucede con bastante frecuencia, que deba repetirse un cálculo o realizar una misma tarea con distintos datos. Sería una manera ilógica, en principio, editar las instrucciones de cálculo tantas veces como datos tengamos, ya que se utilizaría más memoria del ordenador; una forma algo más lógica sería utilizar un *bucle* (loop en inglés), que repitiera las mismas instrucciones tantas veces como quisieramos.

En programación, la terminología inglesa de los tipos de bucle más utilizada es:

- DO WHILE
- REPEAT UNTIL
- FOR ... NEXT

DO WHILE

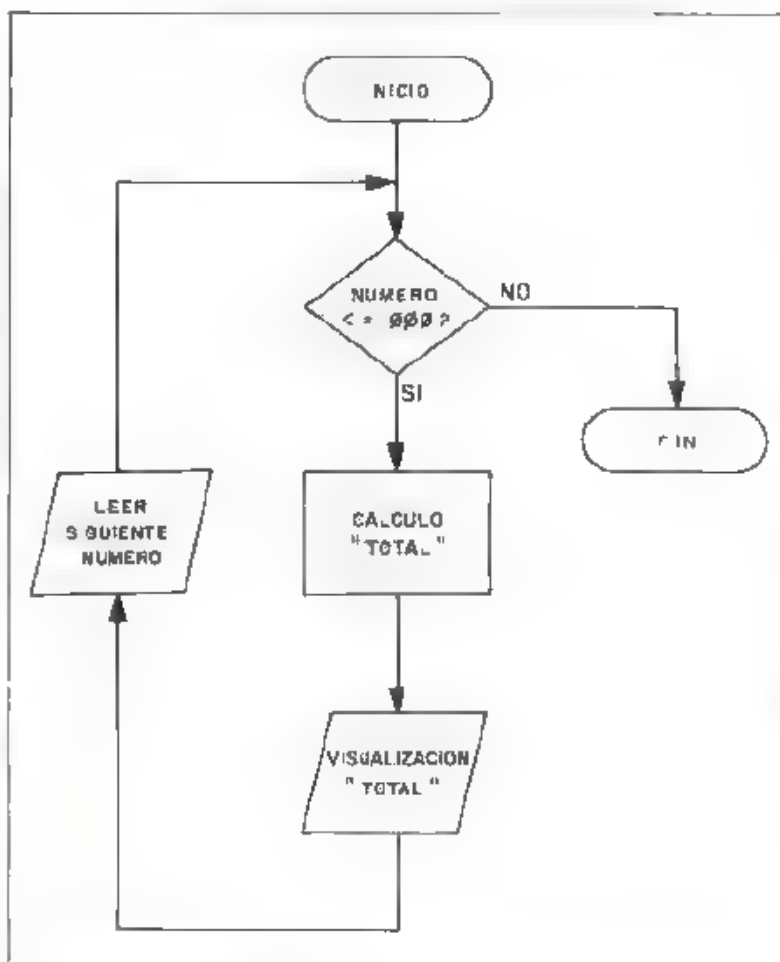
La estructura de bucle DO WHILE (Hacer mientras...) permite realizar una tarea varias veces, siempre y cuando se cumpla la condición impuesta.

Ejemplo:

```
10 REM *****
11      DO WHILE
12      *****
13
14 LET total=0
15 INPUT "Numero > "; numero
16 IF numero > 1000 THEN PRINT
17      "Fin"
18      STOP
19 LET total=total+numero
20 PRINT "Numero: ", numero
21 GO TO 15
```

mientras se cumpla que la variable «numero» sea igual o inferior a «1000», se realizarán indefinidamente las tareas de cálculo y visualización de la variable «total».

108 MICROBASIC



Estructura «DO WHILE».

REPEAT UNTIL

Con este tipo de bucle, (Repetir hasta..., es la traducción) el ordenador realiza una y otra vez la tarea hasta que se da una condición.

Ejemplo

```
10 REM *****
11      DO UNTIL
12      *****
13
14 LET total=0
15 INPUT "Numero > "; numero
16 LET total=total+numero
17 PRINT "Numero: ", numero
18 IF total > 1000 THEN PRINT
19      "Fin"
20      STOP
21 GO TO 15
```

en este otro ejemplo, se repite el bucle hasta que se cumple la condición de que la variable «total» es superior a «1000».

Diferencias

La diferencia entre estos dos tipos de estructura estriba, en que en el primer caso (DO WHILE) la salida del bucle se encuentra antes de realizar la tarea, y en el segundo (REPEAT UNTIL) se encuentra al final. ¿Qué significado prác-

tico tiene esto? Que con una estructura DO WHILE, si al entrar en el bucle no se cumple la condición prevista, se sale de él sin haber ejecutado ninguna tarea; sin embargo, con la estructura REPEAT UNTIL, al menos *una vez* se ejecutan las instrucciones contenidas en él.

FOR/NEXT

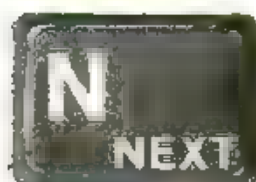
Acceso al teclado

SGN



MODO **K**

IN KEY \$



MODO **K**

OVER

Tipo de sentencia

Comando de programación.

Definición

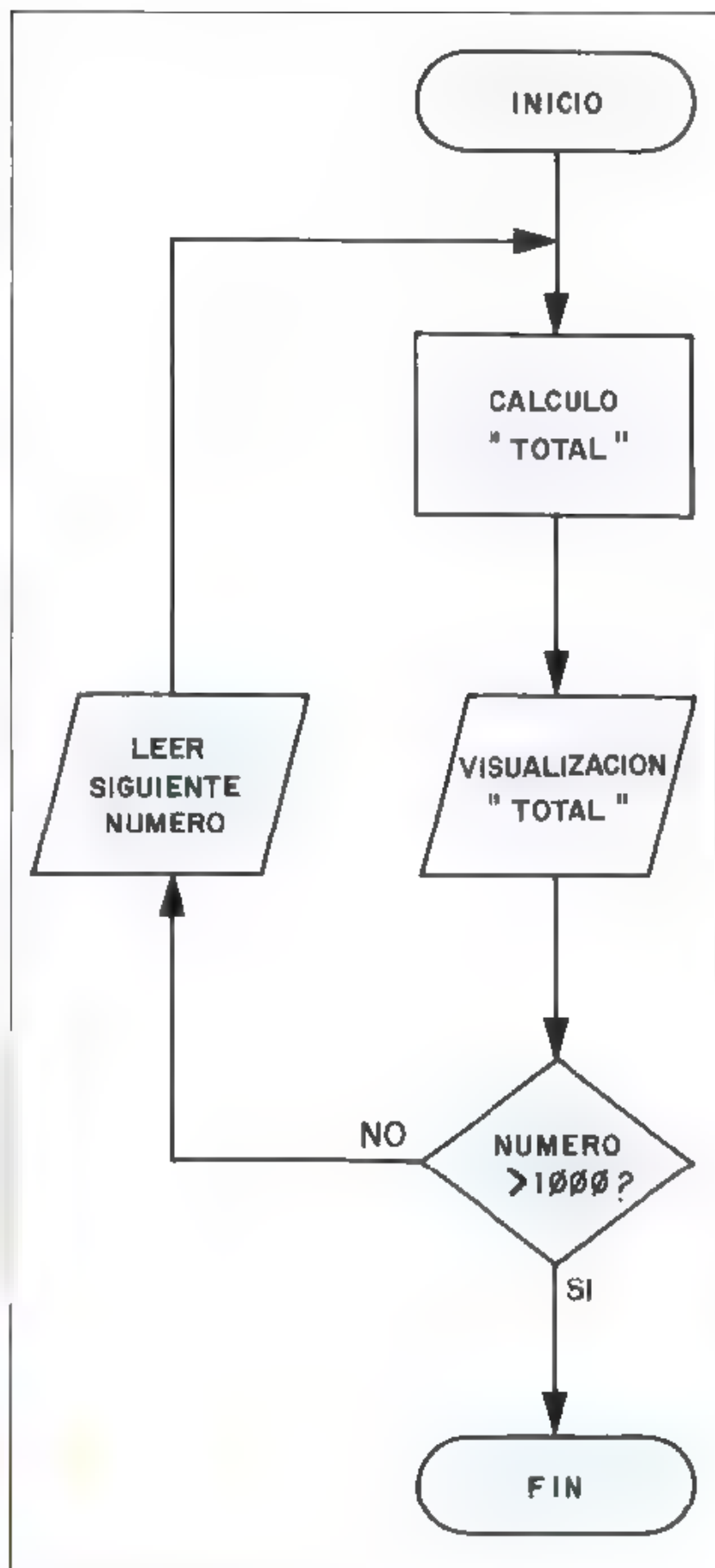
Los bucles «FOR/NEXT» permiten realizar las instrucciones contenidas en él tantas veces como se indica en los límites del argumento. El comienzo del bucle está definido por la sentencia «FOR», y el final por «NEXT». La estructura de la sentencia «FOR» es.

SENTENCIA	ARGUMENTO
FOR	var.= exp.TO exp.

var. = variable.

exp. = expresión.

dentro del argumento se utili-



Estructura «REPEAT UNTIL».

za la palabra clave «TO», cuyo acceso al teclado es:



SGN



La estructura de «NEXT» tiene el siguiente formato:

SENTENCIA	ARGUMENTO
NEXT	variable

Ejemplos:

```
10 FOR J = 2 TO 10
```

```
50 NEXT J
```

En palabras sencillas, la explicación del ejemplo anterior es:

Ejecutar las instrucciones siguientes al «FOR ... TO ...» hasta que la variable «J», partiendo de «2», tome el valor «10»; la sentencia «NEXT» se encarga de incrementar el valor de esta variable; el bucle se repite por tanto, nueve veces.

La variable de control del bucle, sólo puede estar formada por una letra, si existe otra variable con el mismo nombre, ésta es borrada y asume el nuevo valor. La expresión anterior al «TO» es el valor inicial que debe tomar la variable de control, y la expresión posterior el valor final. A diferencia de otros lenguajes BASIC, la

sentencia «NEXT» debe incluir como argumento el nombre de la variable de control; por tanto *no puede omitirse*.

Ejemplo:

```
10 REM *****
   FOR . NEXT
*****
20 FOR N=1 TO 61
30 PRINT "MICROHOBBY "
40 NEXT N
```

En el programa anterior la cadena alfanumérica «MICROHOBBY» es visualizada 61 veces; se utiliza para formatear, el signo ortográfico «;».

La variable de control puede ser incluida en el grupo de sentencias que forman el bucle.

Ejemplo:

```
10 REM *****
   C CUADRADOS
*****
20 FOR C=100 TO 199
30 PRINT "Numero "; C
40 LET A=C^2
50 PRINT "Cuadrado "; A
60 NEXT C
```

la variable de control «C» que varía entre 100 y 199 es elevada al cuadrado, asignando este valor a la variable «A»; ambas variables son visualizadas.

Los límites, al ser expresiones de tipo numérico, pueden estar constituidos por variables previamente asignadas.

Ejemplo:

```
10 REM *****
   PARAMETROS
*****
20 INPUT "Límite inferior ";
```

```
30 INPUT "Límite superior ";
SUPERIOR
40 IF SUPERIOR<INFERIOR THEN G
TO 20
50 FOR A=INFERIOR TO SUPERIOR
60 PRINT "Inferior " IN ERIO
70 PRINT "Superior " SUPERIOR
80 NEXT A
```

En este programa, los parámetros o límites son introducidos con sentencias «INPUT»; la línea 40 verifica que el valor asignado a la variable «superior» es mayor que el de la variable «inferior», en caso contrario, han de introducirse, de nuevo, los límites.

Dentro de un bucle se puede modificar el valor de la variable de control.

Ejemplo:

```
10 REM *****
   MODIFICAR 1
*****
20 FOR X=10 TO 40
30 LET X=X-2
40 PRINT "Valor de X "; X
50 NEXT X
```

Como se puede observar, al ejecutar el programa, la variable «X» no alcanza los valores asignados en los límites, ya que en la línea 30 se decrece en dos su valor, ¿por qué al visualizar «X» se decrece sólo en una unidad?, la respuesta es sencilla, ya que la instrucción «LET X = X - 2» decrece dos unidades, pero la sentencia «NEXT X» incrementa en uno su valor, luego $-2 + 1 = -1$.

Cuando se modifican los límites en el interior de un bucle, el intérprete BASIC hace caso omiso de los nuevos valores y lo ejecuta con los iniciales.

Ejemplo

```

10 REM *****
   MODIF 2
*****
20 LET LIM INF=1
30 LET LIM SUP=20
40 FOR I=1 TO LIM SUP
50 LET I=10 I=30
60 LET I=50
70 PRINT CONTROL " I, Limite
   LIM INF y LIM SUP
80 NEXT I

```

A pesar de las líneas 50 y 60 que modifican los límites, la variable «I» asume los valores iniciales 1 a 20.

Dentro de un bucle «FOR/NEXT» se puede incluir una instrucción del tipo «IF ... THEN ...» que si se cumple la condición, se produzca una ruptura en su ejecución, aunque la variable de control no haya alcanzado su valor final.

Este tipo de estructura de bucle es similar a la que, en otros lenguajes de más alto nivel, se denomina «DO WHILE/BREAK».

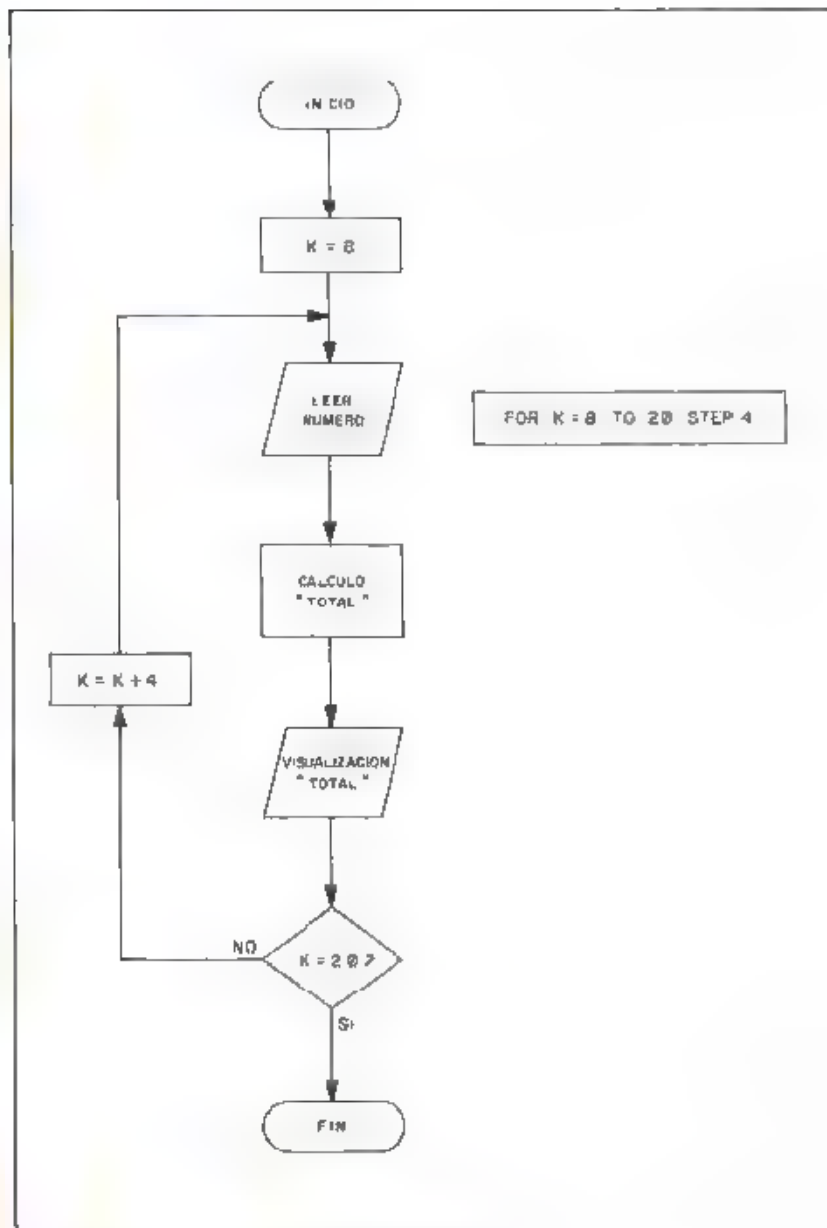
Ejemplo:

```

10 REM *****
   BREAK
*****
20 FOR I=1 TO 15
30 INPUT "Rad 0 >>> " Rad0
40 IF Rad0=0 THEN GO TO 60
50 LET Area=PI*Rad0^2
60 PRINT "Radio " Rad0, "Area " Area
70 NEXT I
80 PRINT "*****"

```

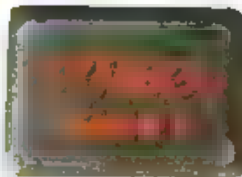
Si se introducen valores de radio distintos de «0», el bucle se ejecuta un número de veces que coincide con los parámetros especificados (15 veces); si por el contrario, se introduce el código de ruptura, que en este caso es igual a «0», la ejecución pasa a la línea 10 sin haber concluido el bucle.



Desarrollo de un bucle FOR-NEXT.

STEP

Acceso al teclado



DATA

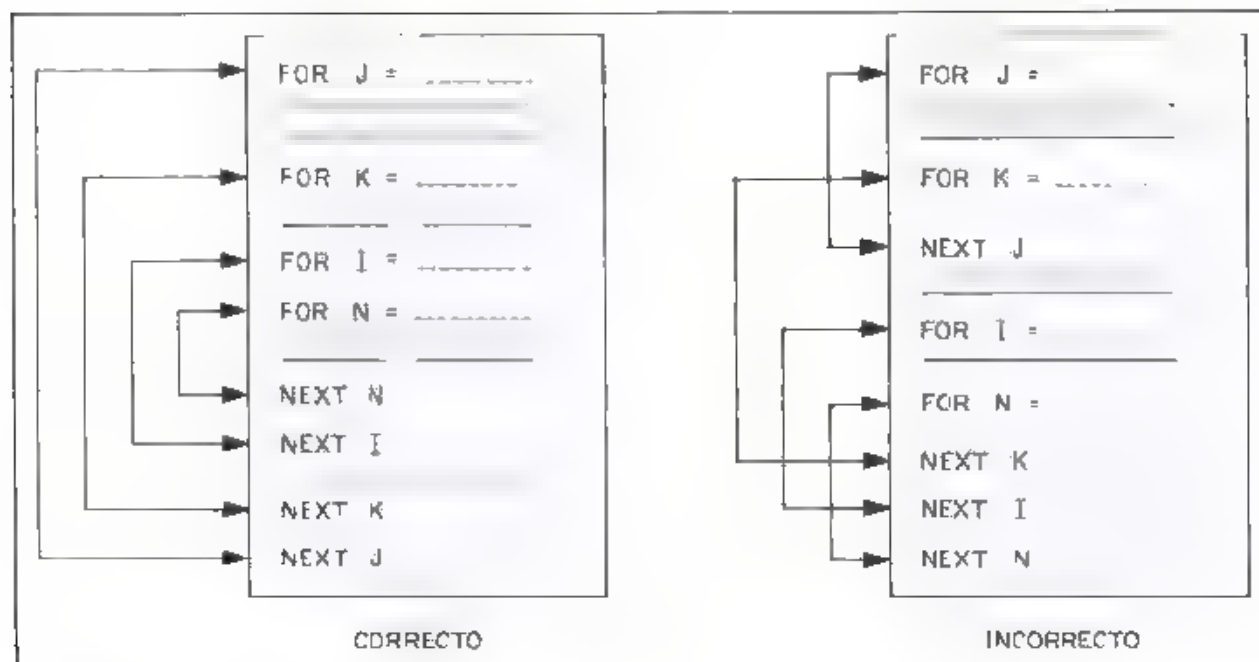


Tipo de sentencia

Comando de programación.

Definición

La palabra clave «STEP» se maneja conjuntamente con la sentencia «FOR ... TO ...», formando parte de su argumento. Se utiliza para modificar el paso de un bucle, es decir, el incremento que se realiza a la variable de control cuando se ejecuta la sentencia «NEXT» correspondiente.



Anidamiento de bucles.

Cuando «STEP» se omite, el intérprete BASIC toma por defecto el valor «1», como se ha visto en los ejemplos anteriores, que la variable de control asumía inicialmente el valor del límite inferior y se incrementaba en uno hasta alcanzar el del límite superior.

Su estructura es la siguiente:

SENTENCIA	ARGUMENTO
FOR	.. TO STEP expresion

Ejemplo:

```
FOR 8 TO 32 STEP 4
```

la variable de control «i» tomará el valor «8» y se incrementará de cuatro en cuatro hasta alcanzar el valor «32», es decir, 8-12-16-20 ... 32. El bucle se realiza siete veces.

Ejemplo:

```
10 REM *****
   STEP *****
20 INPUT PASO >>> PASO
30 FOR I=8 TO 32 STEP PASO
40 PRINT "Variable de control"
50 NEXT I
```

el número de veces que se ejecuta el programa anterior, así como los valores que toma la variable de control «n», depende del que se asigne a la variable «paso», introducida por el teclado.

El paso puede ser una expresión negativa, en este caso, el valor inicial debe ser superior al final.

Ejemplo:

```
10 REM *****
   NEGATIVO *****
15 LET BUC=22222
16 LET EXIJO=80
20 INPUT "Va de 10 a 120"
30 INPUT "2 3 4 para"
40 IF BUC=10 THEN LET I=10
50 GO TO 9 FOR
60 INPUT PASO
70 IF PASO=10 THEN LET I=80
80 GO TO 9 FOR
90 PRINT "OK"
100 PAUSE 50
110 GO TO LINEA
```

este programa sólo se ejecuta con bucles decrecientes, ya que presenta un mensaje de

ERROR si el valor inicial es inferior al final o si el paso es mayor o igual a cero.

Bucles anidados

Se denomina bucle *anidado*, aquel que contiene otro en su interior; no existe límite en cuanto a la cantidad de bucles que se pueden anidar.

Los bucles deben estar completamente *encajados*, por lo que no deben solaparse.

Ejemplo

```
10 REM *****
   ANIDADO *****
20 FOR a=1 TO 5
30 FOR b=1 TO 10
40 PRINT "Bucle 1"
50 PRINT "Bucle 2"
60 NEXT b
70 PRINT "*****"
80 NEXT a
```

El bucle interior «b» se ejecuta 50 veces aunque sus límites van del 1 al 10, ya se multiplica por los del exterior «a». El programa visualiza los valores de las dos variables de

control, la variable «a» se incrementa en 1 cada vez que «b» completa su ciclo.

El siguiente programa calcula las potencias segunda, tercera, cuarta y quinta correspondientes a los veinte primeros números:

```
10 REM *****
11      POTENCIACIÓN
12 *****
13 FOR a=1 TO 20
14   LET b=1
15   FOR j=2 TO 5
16    LET pot=1
17    IF a=1 THEN PRINT TAB 0
18    TAB 3; pot; a; GO TO 20
19    PRINT TAB 1; pot; a
20    LET b=b*a
21   NEXT j
22 NEXT a
```

en la primera columna, aparecen los números del 1 al 20 y en las siguientes, por orden, la correspondiente potenciación.

Errores

Hay una serie de mensajes de error relacionados con los bucles «FOR ... NEXT ...»:

- Sentencia «NEXT» sin «FOR».
- El mensaje:

1 NEXT without FOR

aparece cuando el ordenador encuentra una sentencia «NEXT» sin haber ejecutado con anterioridad su sentencia «FOR» correspondiente y, además, existe definida una variable con el mismo nombre que el argumento de «NEXT».

Ejemplo:

```
10 REM *****
11      ERROR 1
12 *****
13 LET j=30
14 LET s=30
15 PRINT s;
16 NEXT j
```

- Variable no encontrada.

2 Variable not found

similar al error anterior, pero sin estar definida ninguna variable con el mismo nombre.

Ejemplo:

```
10 REM *****
11      ERROR 2
12 *****
13 LET s=30
14 PRINT s;
15 NEXT j
```

- Sentencia «FOR» sin «NEXT».

1 FOR without NEXT

Este mensaje lo presenta el ordenador cuando se encuentra con una sentencia «FOR» en la que los límites o el paso están incorrectos y además no encuentra la sentencia «NEXT» correspondiente.

Ejemplo.

```
10 REM *****
11      ERROR 3
12 *****
13 FOR j=1 TO 2
14 LET s=30
15 PRINT s;
```

- STEP 0.
- Cuando por error se edita una sentencia «FOR» con paso 0, la variable de control no se incrementa al ejecutarse a sentencia «NEXT» correspondiente.

Ejemplo:

```
10 REM *****
11      ERROR 4
12 *****
13 FOR j=1 TO 2 STEP 0
14 LET s=30
15 PRINT s;
```

```
20 FOR j=10 TO 200 STEP 0
30 LET s=4+5
40 PRINT Variable Result
50 NEXT j
```

a pesar de estar comprendidos los límites entre 10 y 200, la variable «j» asume el valor «10» una y otra vez y, por tanto, no alcanza el valor final.

- Cuando por error, se omite la palabra clave «STEP» en un bucle decreciente, éste no se ejecuta y, por lo tanto, continúa en la instrucción siguiente al «NEXT». El ordenador no presenta en este caso mensaje de error.

Ejemplo:

```
10 REM *****
11      ERROR 5
12 *****
13 FOR j=200 TO 10
14 LET s=5+5
15 PRINT s;
16 NEXT j
17 PRINT ***** FIN *****
```

Programas

Los dos programas que se muestran a continuación, son aplicaciones de bucles «FOR ... NEXT ...».

El programa número «1» es de utilidad en matemáticas. Según se introducen los datos, se calcula su suma aritmética, su media aritmética y la suma de sus cuadrados.

La estructura es la siguiente:

- 10 : Comentario con el nombre el programa.
- 20 : Asignación de los colores de pantalla.
- 30-50 : Entrada del número de datos a calcular,

PROGRAMA 1

114 MICROBASIC

180-250 : Bucle para la entrada de los datos mensuales.

```

*****
*          *
*  CALCULO  *
*          *
*****

210 LET suma=suma+dato
220 LET cuadrados=cuadrados+dat
0↑2
230 LET media=suma/n
240 REM

*****
*          *
*  BORRADO  *
*          *
*****

250 FOR a=7 TO 13 STEP 2
260 PRINT AT a,19;"
270 NEXT a
280 GO TO 120

```

verificación de los mismos, tienen que estar comprendidos entre «0» y «20» (millones), visualización y salto a la rutina de almacenamiento.

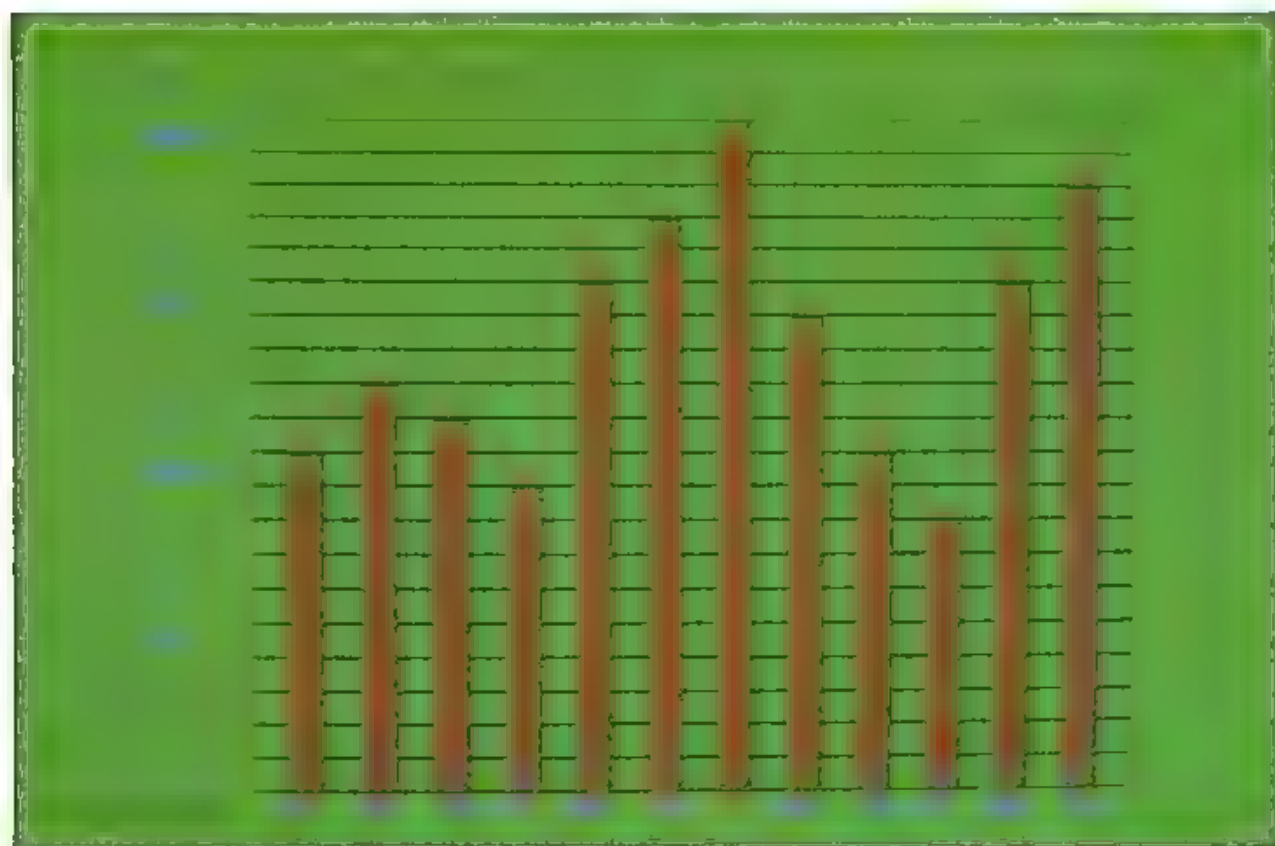
252-256 : Pausa y borrado de la pantalla.
 260 : Salto a la rutina de dibujo.
 262-500 : Almacenamiento de datos y retorno al bucle principal.
 510-560 : Bucle anidado

para dibujar las rayas horizontales del «histograma» con el símbolo del subrayado «_».

570-620 : Visualización de los valores en millones en el eje vertical (tinta azul).

630-680 : Visualización de las iniciales de los meses en el eje horizontal (tinta azul).

690-790 : Bucle para el salto a la rutina de extracción de datos de memoria y su representación en gráfico de barras (tinta roja).
 800 : Salto de la rutina «fin».
 810-1050 : Rutina de extracción de datos y



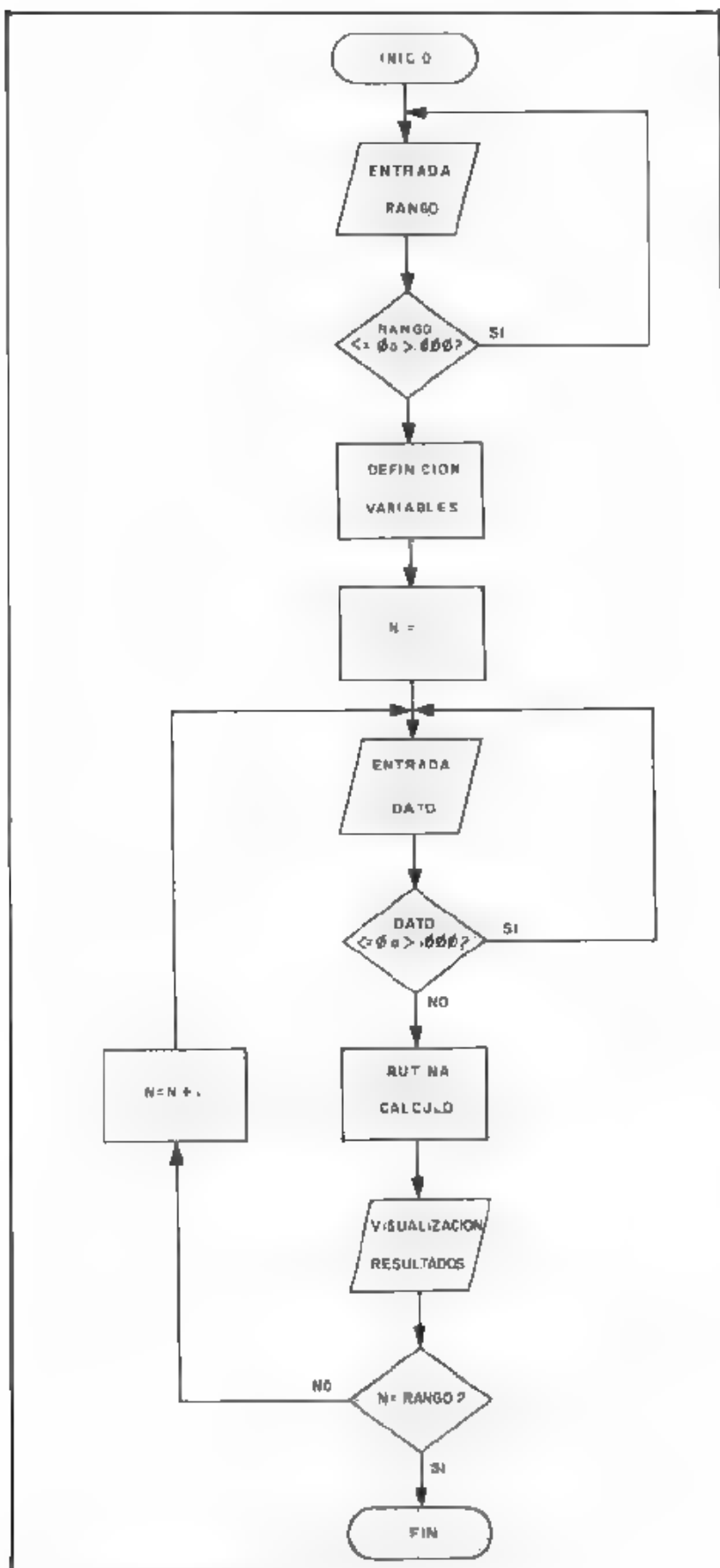
PROGRAMA 2

```

10 REM *****
* HISTOGRAMAS *
*****
20 BORDER 4 PAPER 4 INK 0 C
LS
22 LET memoria=270
30 REM *****
* CARATULA *
*****
40 PRINT AT 0,7;"Entrada de da
tos"
50 PRINT AT 1,7;" "
60 PRINT AT 6,0, enero .....

70 PRINT "Febrero ....."
80 PRINT "Marzo ....."
90 PRINT "Abril ....."
100 PRINT "Mayo ....."
110 PRINT "Junio ....."
120 PRINT "Julio ....."
130 PRINT "Agosto ....."
140 PRINT "Septiembre ....."
150 PRINT "Octubre ....."
160 PRINT "Noviembre ....."
170 PRINT "Diciembre ....."
180 REM *****
* ENTRADA DATOS *
*****
190 FOR n=1 TO 12
200 INPUT "Datos mes ";(n);" >>
dato
210 IF dato<0 OR dato>20 THEN G
O TO 200
220 PRINT AT 5+n,15,dato,AT 5+n
,18,"millones"
230 GO TO memoria
240 LET memoria=memoria+20
250 NEXT n
252 PRINT 40;"Pulse una tecla p
ara continuar"
254 PAUSE 0
256 CLS
260 GO TO 510
262 REM *****
* MEMORIA *
*****
270 LET enero=dato
280 GO TO 240
290 LET febrero=dato
300 GO TO 240
310 LET marzo=dato
320 GO TO 240
330 LET abril=dato
340 GO TO 240
350 LET mayo=dato
360 GO TO 240
370 LET junio=dato
380 GO TO 240
390 LET julio=dato
400 GO TO 240
410 LET agosto=dato
420 GO TO 240
430 LET septiembre=dato
440 GO TO 240
450 LET octubre=dato
460 GO TO 240
470 LET noviembre=dato
480 GO TO 240
490 LET diciembre=dato
500 GO TO 240
510 REM *****
* HORIZONTALES *
*****
515 PRINT 40,AT 1,3;"Espere un
momento, por favor"
520 FOR n=0 TO 20
530 FOR x=4 TO 28
540 PRINT AT n,x;"_"
550 NEXT x
560 NEXT n
570 REM *****
* VALORES *
*****
580 LET paso=20
590 FOR j=1 TO 16 STEP 5
600 PRINT INK 1,AT j,1,paso
610 LET paso=paso-5
620 NEXT j
630 LET a$="EFMMAMJJASOND"
640 LET mes=1
650 FOR j=5 TO 27 STEP 2
660 PRINT INK 1,AT 21,j,a$(mes)
670 LET mes=mes+1
680 NEXT j
690 REM *****
* DIBUJO *
*****
700 LET memoria=820
710 FOR j=5 TO 27 STEP 2
720 GO TO memoria
730 IF valor=0 THEN GO TO 780
740 LET valor=21-valor
750 FOR i=20 TO valor STEP -1
760 PRINT INK 2,AT i,j;"■"
770 NEXT i
780 LET memoria=memoria+20
790 NEXT j
800 GO TO 1060
810 REM *****
* MEMORIA *
*****
820 LET valor=enero
830 GO TO 730
840 LET valor=febrero
850 GO TO 730
860 LET valor=marzo
870 GO TO 730
880 LET valor=abril
890 GO TO 730
900 LET valor=mayo
910 GO TO 730
920 LET valor=junio
930 GO TO 730
940 LET valor=julio
950 GO TO 730
960 LET valor=agosto
970 GO TO 730
980 LET valor=septiembre
990 GO TO 730
1000 LET valor=octubre
1010 GO TO 730
1020 LET valor=noviembre
1030 GO TO 730
1040 LET valor=diciembre
1050 GO TO 730
1060 REM *****
* FINAL *
*****
1070 INPUT "Otros datos? (S/N) >
"; LINE a$
1080 IF a$="s" OR a$="S" THEN GO
TO 10
1090 IF a$="n" OR a$="N" THEN ST
OP
1100 GO TO 1070

```

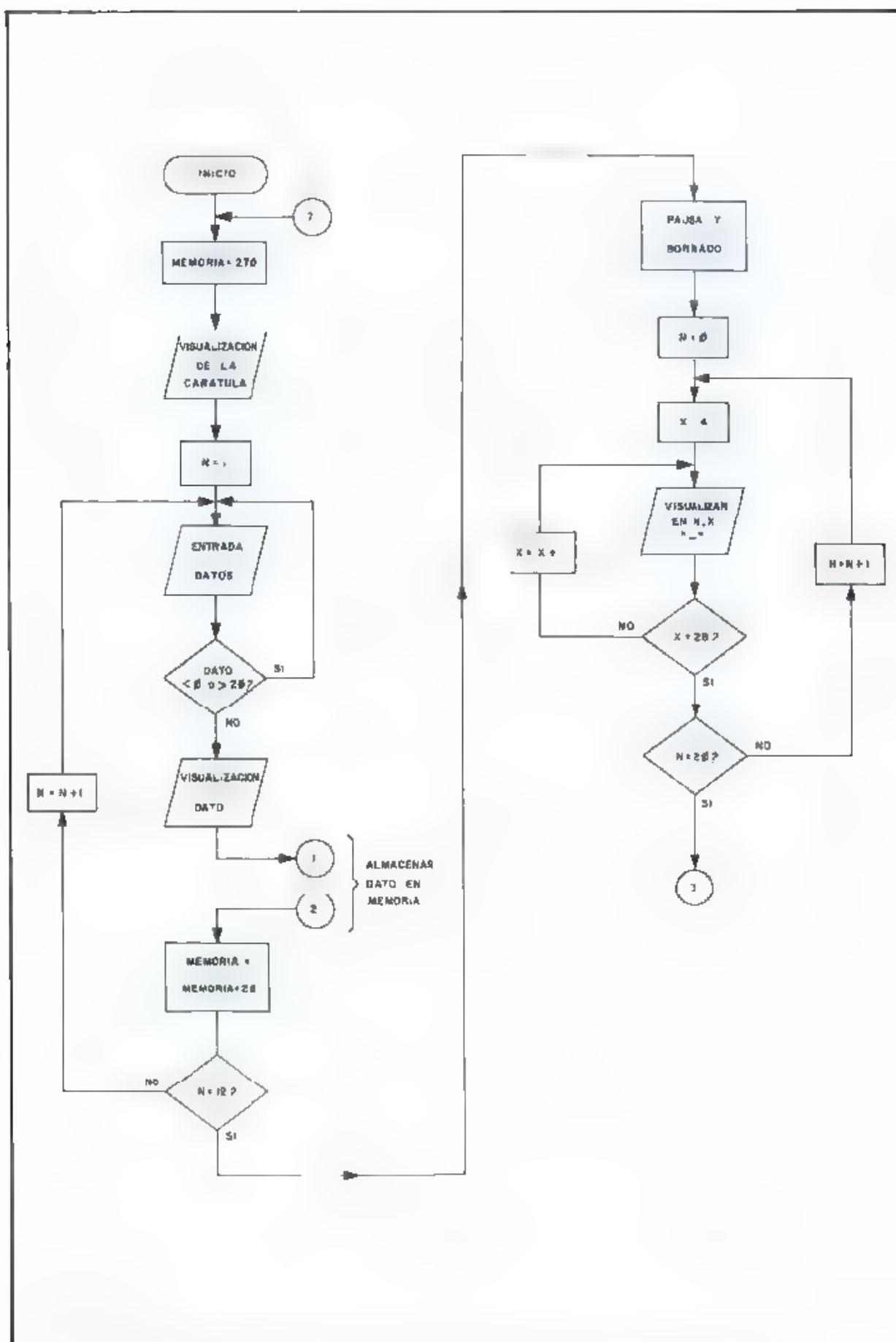



Programa «ESTADISTICA».

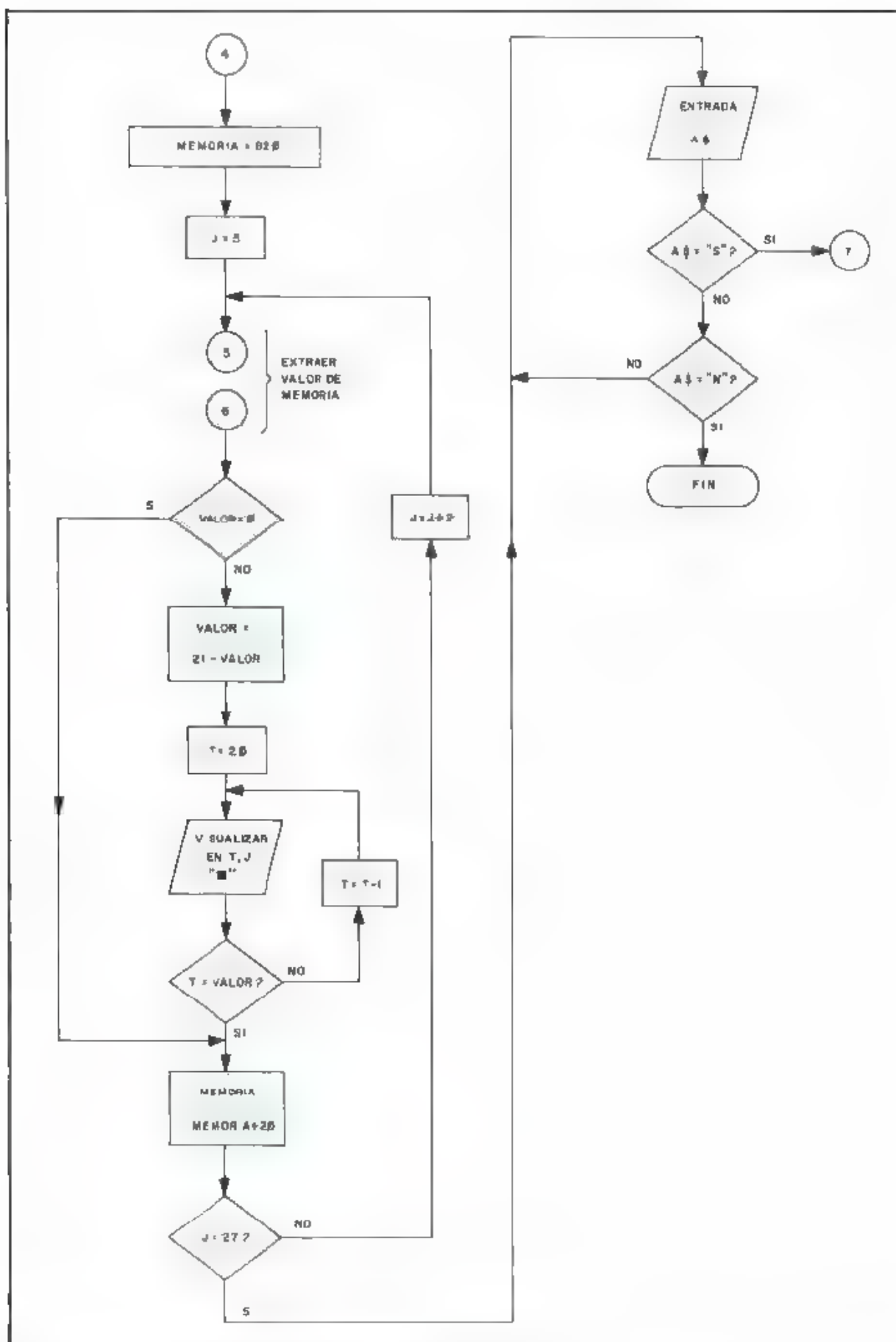
retorno al bucle principal

1050-1100 : Decisión, de retorno al principio o terminar el programa.

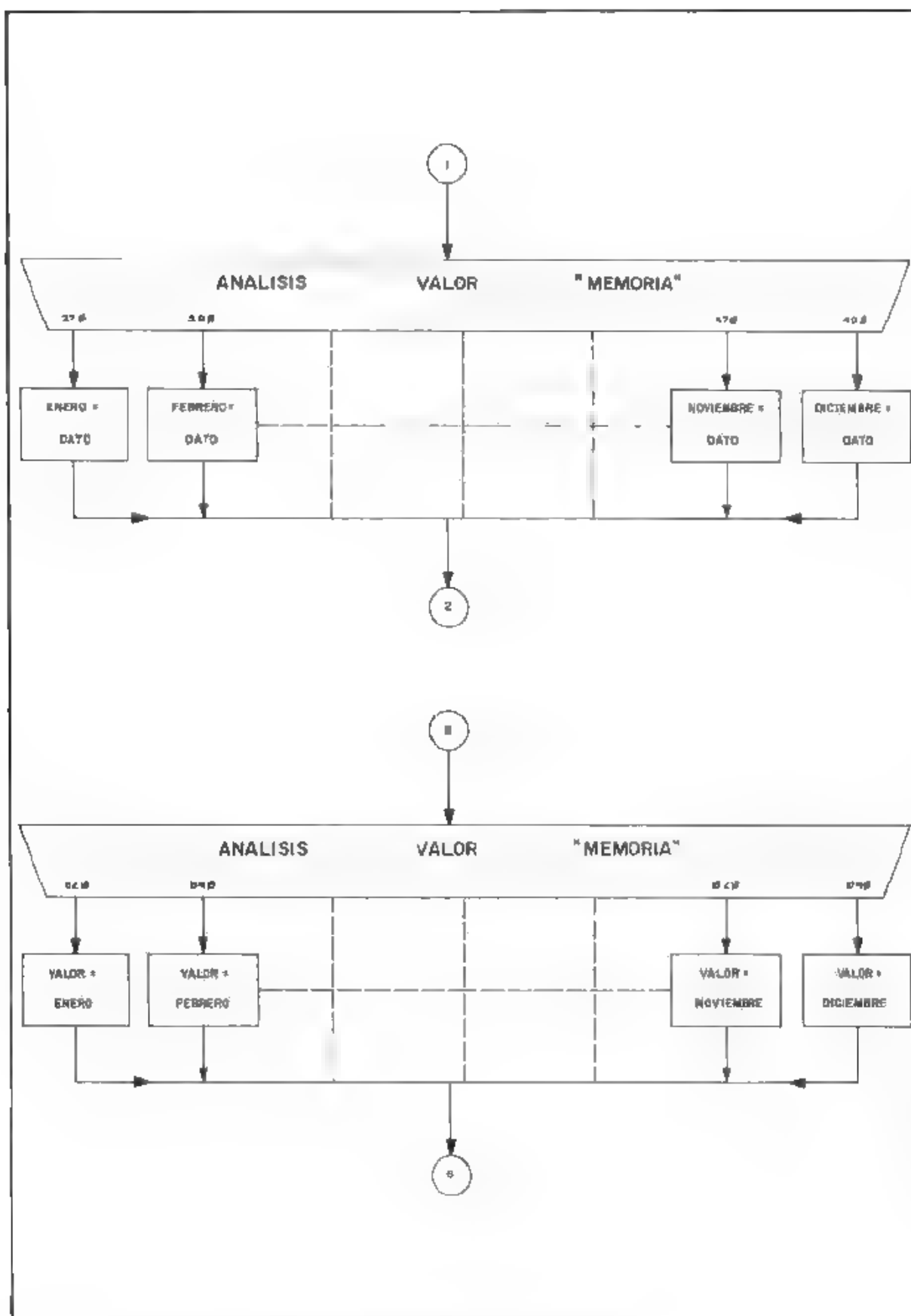
La rutina de almacenamiento y extracción de datos, se ha realizado de una manera un tanto especial, ya que todavía no se han visto las *matrices*, que serán estudiadas en un capítulo posterior. ■



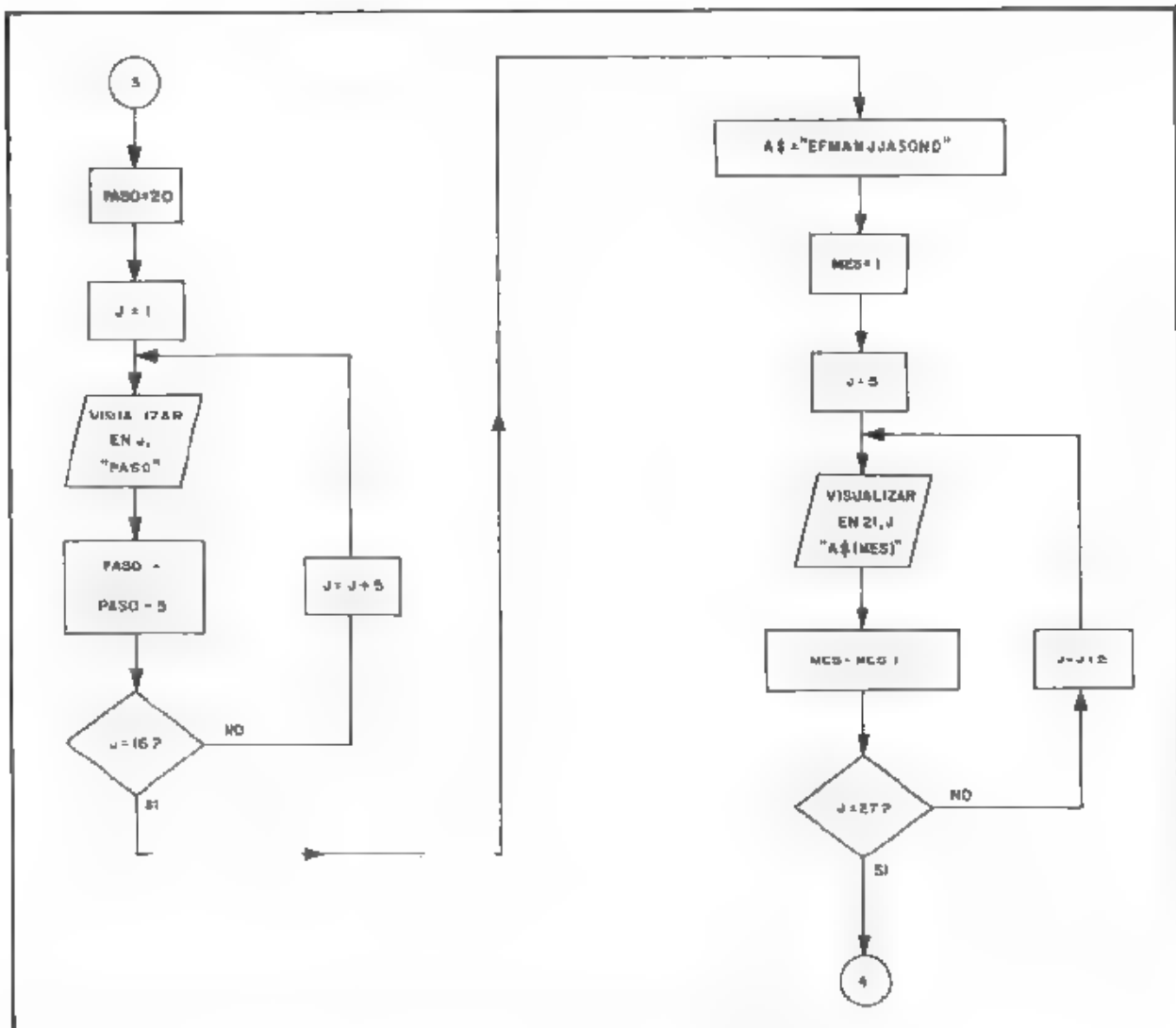
Programa «HISTOGRAMA» entrada de datos y dibujo líneas horizontales.



Programa «HISTOGRAMA» dibujo, barras y rutina «FIN».



Programa «HISTOGRAMA» almacenamiento y extracción de datos.



Programa «Histograma» datos eje vertical y horizontal.

SUBROUTINAS

Normalmente, en un programa hay ciertos cálculos o funciones que se necesitan en *distintas* partes del mismo; en lugar de editarlas varias veces, conviene hacerlo una sola vez en formato *subrutina* o *sub-programa*, de esta manera, este grupo de instrucciones sólo se ejecutan cuando el programa principal lo indica, mediante una instrucción de llamada a subrutina; cuando ésta termina de ejecutarse devuelve el control al programa principal.

Un bucle también repite varias veces una serie de instrucciones, pero siempre en una misma zona de programa; a diferencia, la subrutina puede ser llamada desde cualquier parte.

Dentro de la programación *estructurada*, la utilización de subrutinas es un hábito muy recomendable. Conviene para una mayor estructuración que todas las subrutinas estén localizadas en la zona final del programa; una a continuación de la otra.

GO SUB

Acceso al teclado

SQR



MODULO K

CIRCLE
122 MICROBASIC

Tipo de sentencia

Comando de programación

Definición

Es la instrucción utilizada, dentro de un programa, para «llamar» a una subrutina, aunque también admite el formato de comando directo.

Su estructura general es:

SENTENCIA	ARGUMENTO
GO SUB	N.º de línea

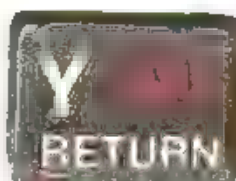
Ejemplos:

- GO SUB 500.
- GO SUB 30.

RETURN

Acceso al teclado

STR S



MODULO K

Tipo de sentencia

Comando de programación

Definición

Esta palabra clave se utiliza para retomar, de una subrutina, al programa principal.

Su estructura es la siguiente:

SENTENCIA	ARGUMENTO
RETURN	_____

Utilización de «GO SUB» y «RETURN»

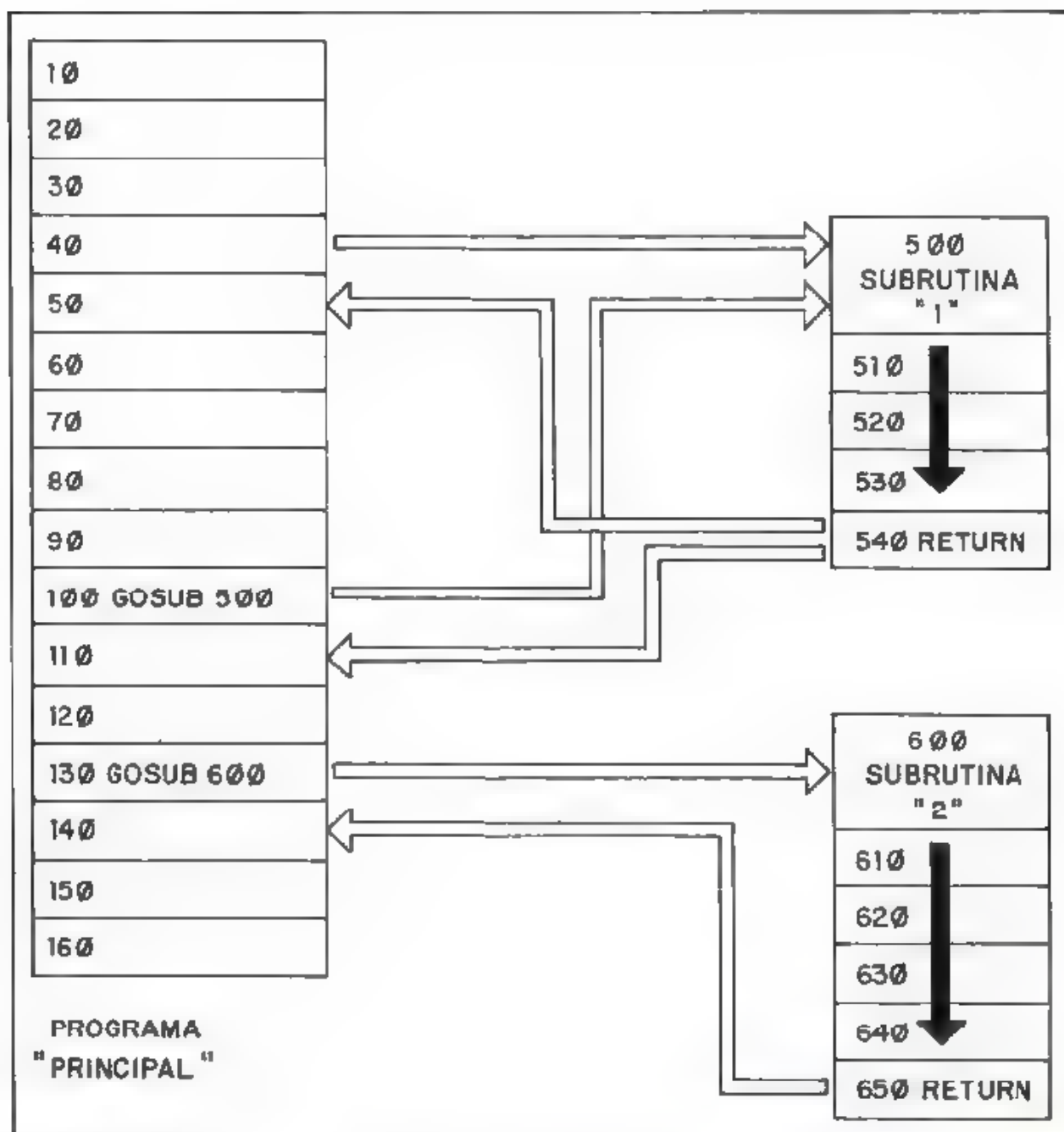
La llamada a subrutina «GO SUB» puede hacerse en cualquier parte del programa que se necesita. La palabra clave «RETURN», sin embargo, debe utilizarse *siempre* al final de cada subrutina.

Ejemplo:

10	REM
		GO SUB
	
20	INPUT	A?
30	INPUT	B?
40	GO SUB	100
50	PRINT	
60	STOP	
100	REM
		SUBROUTINA COMPARA
	
110	IF	a=b THEN LET
120	GO TO	130
130	LET	Z=a-b
140	RETURN	

Cuando se ejecuta la línea «40» el programa principal cede el control a la subrutina localizada en la línea «100», ésta se va ejecutando hasta encontrar la sentencia «RETURN» que causa el retorno al programa principal (línea 50).

A primera vista, parecen similares las sentencias «GO TO» y «GO SUB», ya que ambas provocan un salto al número de línea especificado en



Llamadas a «subrutinas».

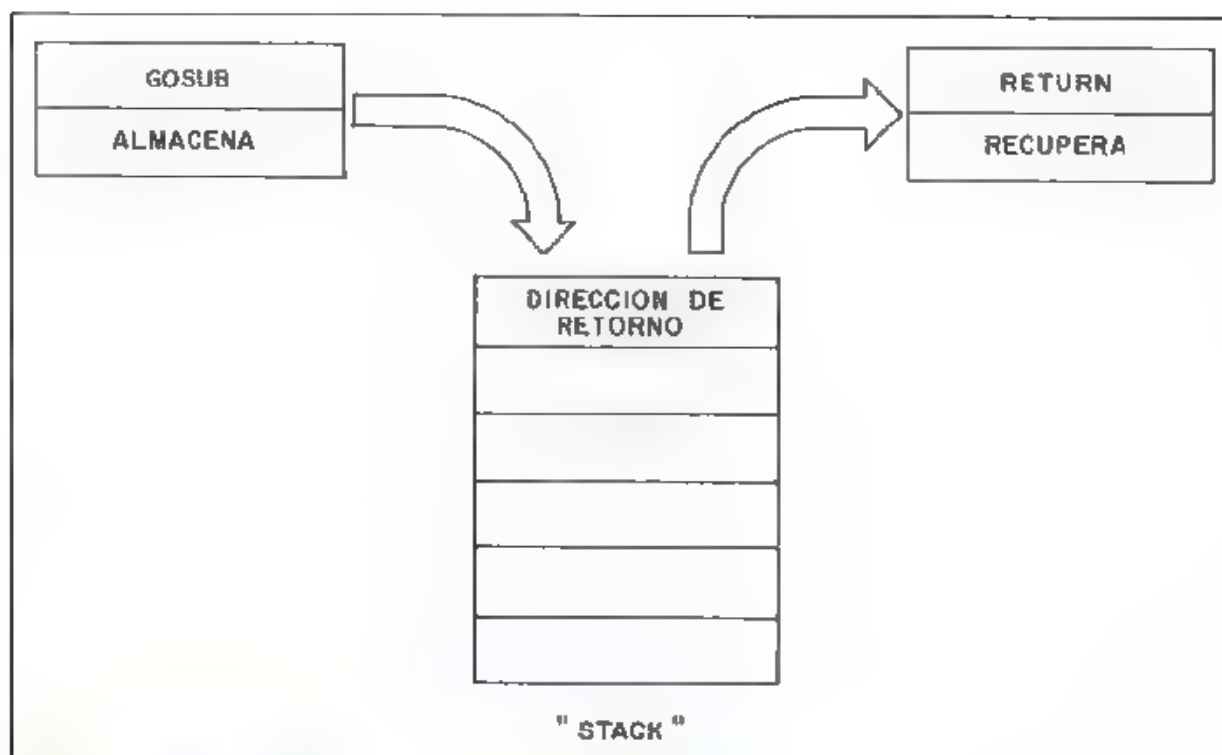
su argumento, la diferencia radica en que, el ordenador, cuando se ejecuta la sentencia «GO SUB», apunta en una zona de la memoria denominada STACK o «Fila de GO SUB», memoria tipo «LIFO» (Last Input First Output) en la que el último dato almacenado es el primero en ser recuperado, la *dirección de retorno*, formada por el número de línea y sentencia dentro de la línea de la

instrucción «GO SUB». Cuando se ejecuta la última línea de la subrutina (RETURN), se recupera la dirección de retorno almacenada en la memoria, y el programa continúa en la instrucción siguiente. Como se puede observar, el usuario no tiene que preocuparse para nada de la dirección de retorno de una subrutina, de esta manera su manejo se hace sencillo.

Las llamadas a subrutina pueden hacerse con variables numéricas, cuyo valor sea el número de línea donde están localizadas.

Ejemplo:

```
10 REM *****
   : PARABLES :
   : *****
20 LET entrada=90
30 LET v SUB STACK=120
```



Pila de «Gosub».

```

40 PRINT Subrutina 1
50 GO SUB 100
60 PRINT Subrutina 2
70 GO SUB 100
80 STOP
90 REM
100 INPUT Nombre : LINE 0
110 INPUT Apellido : LIN
120 RETURN
130 PRINT Tu nombre es : NS
140 REJUN

```

Tipos de subrutinas

Las subrutinas pueden clasificarse según la filosofía de su ejecución en

- Subrutinas sin parámetros.
- Subrutinas con parámetros.

Una subrutina sin parámetros es aquella que realiza un cálculo o función siempre de la misma manera, es decir con los mismos datos o valores.

Ejemplo:

```

10 REM *****
   SIN PARAMETROS
   *****
20 GO SUB 100
30 PRINT AT 10 10 MICROHOBBY
40 STOP
100 REM *****
   SUBROUTINA
   *****
105 CLS
110 FOR A=0 TO 21
120 PRINT PAPER A AT A 0
130 NEXT A
140 RETURN

```

La subrutina localizada en la línea «100», borra y posteriormente coorea de verde la pantalla, siempre que es llamada.

Las subrutinas con parámetros se ejecutan de forma más generalizada, ya que permiten realizar una misma tarea para diversos valores.

El siguiente programa es una variable del anterior, permite borrar y colorear la pantalla del color que se desee.

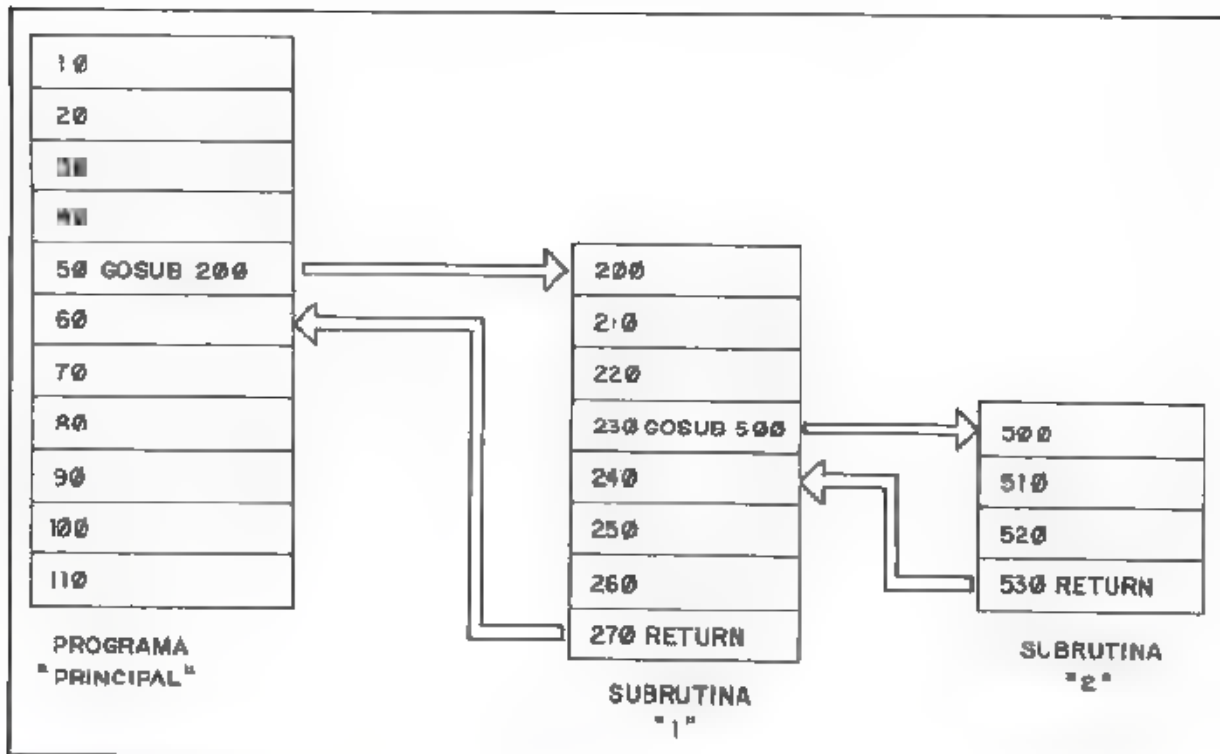
Antes de llamar a la subrutina debe asignarse, a la variable utilizada, el valor correspondiente al color.

```

10 REM *****
   CON PARAMETROS
   *****
15 LET CO=2
20 GO SUB 100
30 PRINT AT 10 10, MICROHOBBY
40 LET CO=4
50 GO SUB 100
60 PRINT AT 10 11, SENARAL
70 GO SUB 100
80 LET CO=5
90 GO SUB 100
100 PRINT AT 0 10 MICROHOBBY
110 PRINT AT 10 11 SENARAL
120 STOP
100 REM *****
   SUBROUTINA
   *****
105 FOR A=0 TO 21
110 PRINT PAPER CO AT A 0
120 NEXT A
130 RETURN
140 FOR A=0 TO 150
150 NEXT A
160 RETURN

```

Se combina con otra subrutina localizada en la línea «200», que permite ver los mensajes sucesivos, ya que su



«Subrutinas» anidadas.

Única misión es la de *temporizar*.

Los parámetros pueden ser tanto de *entrada* como de *salida*, veamos un ejemplo en el que se combinan ambos.

```

10 REM *****
11 REM : ENTRADA/SALIDA :
12 REM *****
20 INPUT "Precio : " Precio
30 INPUT "Impuesto de Lujo : " Impuesto
40 PRINT "Precio : " Precio
50 PRINT "Impuesto de Lujo : " Impuesto
60 PRINT "Total : "
70 GO SUB 100
80 PRINT "Total : "
90 STOP
100 REM *****
101 REM : SUBROUTINA :
102 REM *****
110 LET Total=Precio+Impuesto
120 LET Sa="Total: ",Total,"Impuesto:"
130 RETURN
  
```

Atendiendo al tipo de llamada, las subrutinas pueden clasificarse en:

INCONDICIONALES. — CONDICIONALES.

Incondicionales son aquellas que no necesitan de ninguna condición para que se ejecuten.

Ejemplo:

```
GO SUB 750
```

Una llamada a subrutina condicional, es aquella que necesita que se cumplan una o varias condiciones previas.

Ejemplo:

```
IF a$ = "PEREZ" THEN GO SUB 100
```

Subrutinas anidadas

Al igual que los bucles, las subrutinas pueden anidarse, es decir, que desde una subrutina se puede llamar a otra y así sucesivamente. Las direcciones de retorno se van almacenando, como ya se comentó anteriormente, en el STACK.

la primera dirección en recuperarse corresponde con la de la última subrutina llamada.

Aunque no es muy frecuente, una subrutina puede llamarse a sí misma, esto es lo que en programación se denomina *subrutina Recursiva*.

Un ejemplo de subrutinas anidadas es el siguiente:

```

10 REM *****
11 REM : ANIDADAS :
12 REM *****
20 LET Fondo=4
30 LET Total=0
40 GO SUB 100
50 PRINT "Fondo: " Fondo
60 STOP
70 REM *****
71 REM : SUBROUTINA :
72 REM *****
110 FOR a=0 TO 12
120 FOR b=0 TO 23
130 PRINT "Fondo: " Fondo
140 IF a=0 OR b=11 THEN GO SUB 200
150 GO TO 160
160 NEXT b
170 RETURN
200 PRINT "Fondo: " Fondo
210 RETURN
  
```

PROGRAMA 1

```

10 REM *****
   *      CURSO/BASIC      *
   *      *****      *
   *      ADIVINO      *
   *      *****      *
12 BORDER 4: PAPER 4: INK 1: CLS
14 REM *****
   *      VARIABLES      *
   *      *****      *

16 LET record=99
20 LET instruccion=1000
30 LET error=1100
40 LET temporizacion=1200
50 LET calculo=1300
60 LET verificacion=1400
70 LET acierto=1500
80 LET seguir=1600
100 REM *****
   *      BUCLE CENTRAL      *
   *      *****      *

102 LET intento=1
104 LET mayor=101: LET menor=0
106 PRINT AT 4,7,"PROGRAMA "AD
   DIVINO"
110 INPUT "Desea conocer las in
   strucciones (S/N) >>>"; LINE a$
120 IF a$="s" OR a$="S" THEN GO
   TO 160
130 IF a$="n" OR a$="N" THEN GO
   TO 160
140 GO SUB error
150 GO TO 110
160 PRINT AT 9,5,"Pense un num
   ero entre",AT 12,10,"0" y "1
   00"
170 LET retardo=400
180 GO SUB temporizacion
190 CLS
200 PRINT AT 9,3,"Su numero es
   el"
210 LET retardo=100
220 GO SUB temporizacion
230 GO SUB calculo
240 PRINT numero
250 IF igual=1 THEN GO SUB acie
   rto: GO TO 310
255 PRINT "2"
260 INPUT "M=Mayor/N=Menor/I=Ig
   ual >>>"; LINE a$
270 GO SUB verificacion
280 IF reconocimiento=0 THEN GO
   TO 260
290 IF reconocimiento=1 THEN PR
   INT AT 9,25,"LET intento=
   intento+1: GO TO 200
300 IF reconocimiento=2 THEN GO
   SUB acierto
310 GO SUB seguir
320 ELS
330 GO TO 100
999 REM

```

```

*****
*      SUBROUTINAS      *
*      *****      *

1000 REM INSTRUCCION:
1010 PRINT AT 8,3,"Debe pensar u
   n numero entero y positivo, comp
   rendido entre 0 y 100. El ordena
   dor tratara de adivinarlo en el
   menor numero de intentos. Para f
   acilitarle la tarea, debe darle
   alguna pista"
1020 PRINT AT 16,8,"M si es mayo
   r"
1030 PRINT AT 18,8,"N si es meno
   r"
1040 PRINT AT 20,8,"I si es igua
   l"
1050 LET retardo=1200
1060 GO SUB temporizacion
1065 CLS
1070 RETURN
1100 REM ERROR
1110 PRINT AT 1,4,"Respuesta
   no reconocida"
1115 LET retardo=200
1120 GO SUB temporizacion
1130 RETURN
1200 REM TEMPORIZACION
1210 FOR x=1 TO retardo
1220 NEXT x
1230 RETURN
1300 REM CALCULO
1310 LET numero=INT (menor+mayo
   r)/2
1320 IF mayor-menor<2 THEN LET
   igual=1: RETURN
1325 LET igual=0
1330 RETURN
1400 REM VERIFICACION
1410 IF a$="M" OR a$="m" THEN LE
   T menor=numero: GO TO 1460
1420 IF a$="N" OR a$="n" THEN LE
   T mayor=numero: GO TO 1460
1425 IF a$="I" OR a$="i" THEN LE
   T reconocimiento=2: RETURN
1430 GO SUB error
1440 LET reconocimiento=0
1450 RETURN
1460 LET reconocimiento=1
1470 RETURN
1500 REM ACIERTO
1510 PRINT AT 12,8,"ACERTE!"
1520 PRINT AT 15,8,"En",intento,
   " intentos"
1530 LET retardo=400
1540 GO SUB temporizacion
1550 CLS
1560 IF intento<record THEN LET
   record=intento
1570 PRINT AT 9,7,"Mi record est
   a en"
1580 PRINT AT 13,14,"""",record,
   ""
1590 RETURN
1600 REM SEGUIR
1610 INPUT "Quiere jugar otra ve
   z (S/N)"; LINE a$
1620 IF a$="S" OR a$="s" THEN RE
   TURN
1630 IF a$="N" OR a$="n" THEN ST
   OP
1640 GO SUB error
1650 GO TO 1610

```

Error

Es necesario separar las subrutinas del resto de programa mediante sentencias del tipo:

- STOP.
- GO TO n.

ya que de lo contrario, podrían ejecutarse sin haber sido llamadas, provocando un error del tipo:

7 RETURN without GO SUB

Ejemplo:

```

10 REM *****
   *      ERROR      *
   *      *****

```

```

10 LET S=0: A=100
20 INPUT "¿Cuánto? "; A
30 INPUT "Duración "; B
40 GO SUB SMD: A=0
100 REM
      *****
      SUBROUTINE
      *****
110 PRINT "A = " A
120 LET P=0: A=A+B
130 DO: P=P+1: A=A+B
140 NEXT P

```

Programas

Como aplicación a las «subrutinas», se presentan dos programas:

- ADIVINO.
- LONGITUD.

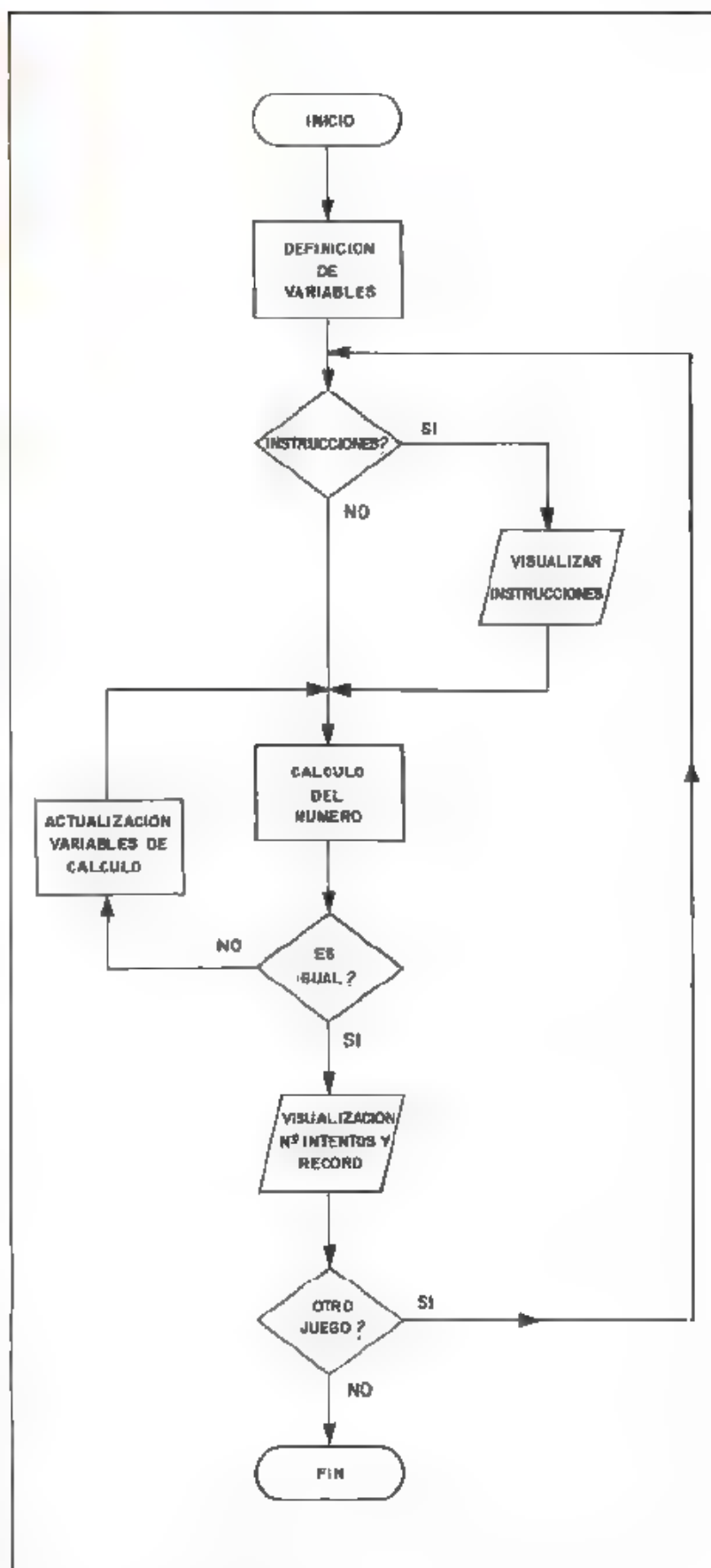
El primero se trata de un juego de *adivinanza*, en el que el ordenador debe descubrir el número pensado por usted, que deberá estar comprendido entre «0» y «100».

Al principio del programa, se da la opción de poder visualizar las instrucciones del juego. El ordenador, posteriormente, presenta un número de pantalla; según la relación que exista entre éste y el número pensado por nosotros, deberemos introducir una de las siguientes claves:

- «M» - Mayor.
- «N» - Menor.
- «I» - Igual.

La estructura general del programa es

- 10 : Comentario con el nombre del programa.
- 12 : Asignación del color verde para el borde y el fondo, y azul para los caracteres.
- 14-80 : Definición de la variable «record», utilizada para almacenar el mínimo número de intentos; inicialmente tiene asignado



Estructura programa «Adivino».

el valor «99», para poder ser actualizado en la primera jugada.

También se definen, para una mejor interpretación, las variables utilizadas como dirección de comienzo de las subrutinas.

102-104 : Inicialización de las variables utilizadas en cada partida.

106-1505 : Presentación del programa y visualización de instrucciones, si se desea.

160-190 : Mensaje de invitación a comenzar el juego.

200-240 : Visualización del número calculado por el ordenador.

250 : Si el ordenador está seguro del número, se ejecuta la subrutina «acierto».

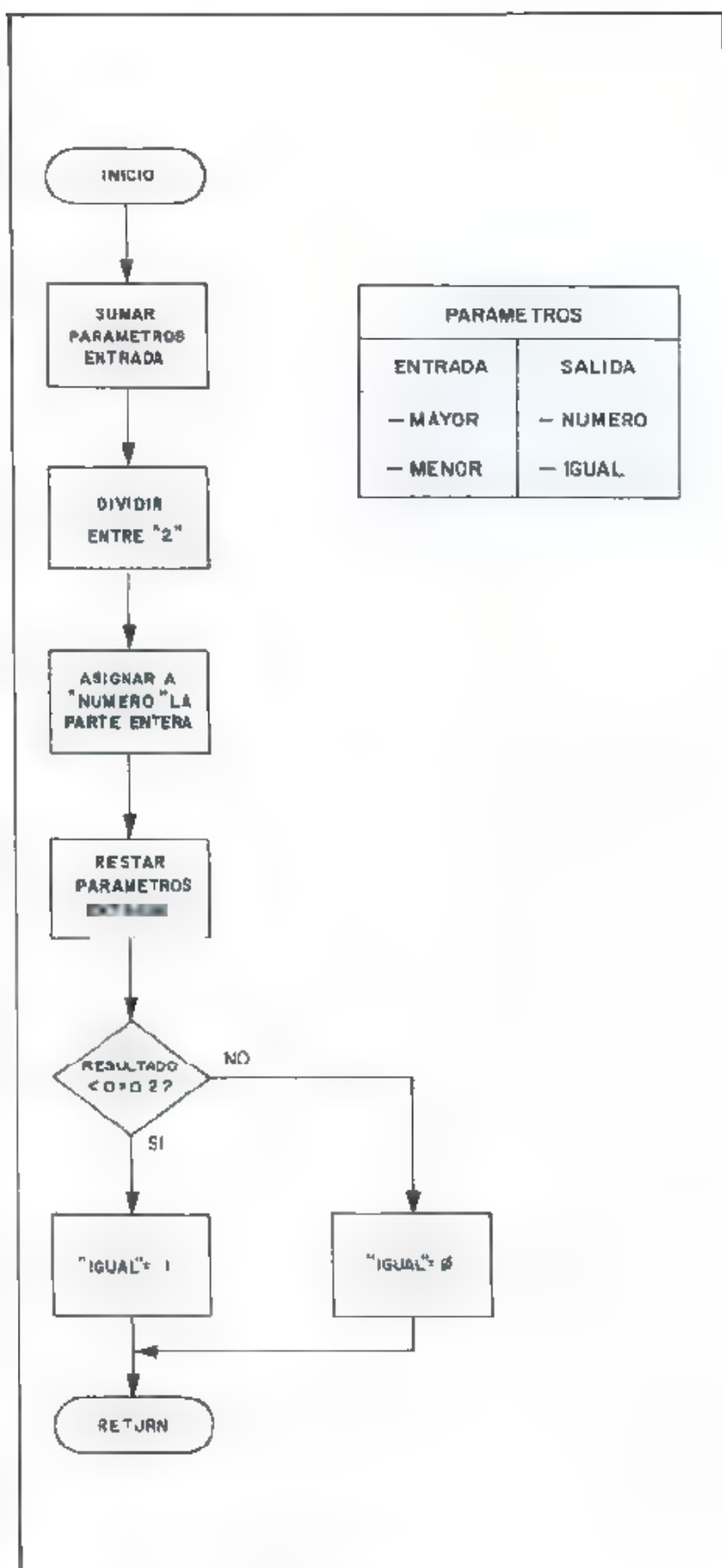
255-290 : Introducción de la clave correspondiente a la pista (M, N o I), si no es «igual», se salta a la línea 200 para calcular un nuevo número.

300 : El número ha sido acertado.

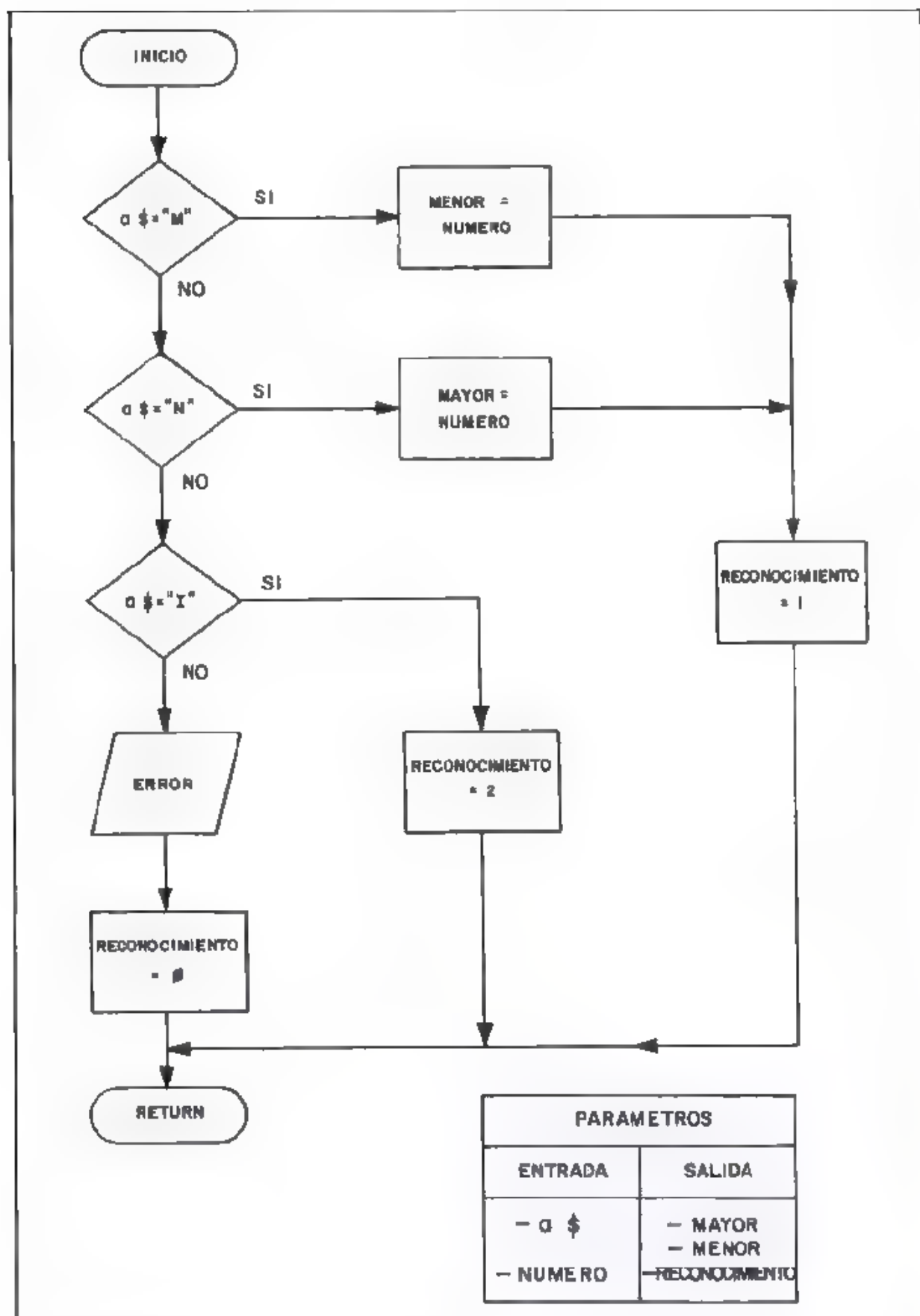
310 : ¿Se desea continuar?

320-330 : En caso afirmativo, se borra a pantalla y se comienza de nuevo en la línea «100».

1000-1070 : Subrutina «INSTRUCCION». Se visualizan las instrucciones durante un tiempo y luego se borra la pantalla. Se llama a la



Programa «Adivino» subrutina «Cálculo».



Programa «Adivino» subrutina «Verificación».

PROGRAMA 2

```

10 REM *****
  * CURSO/RASO *
  * ***** *
  * LONGITUD *
  * ***** *

20 BORDER 4: PAPER 4: INK 1: 0
30 REM

  *****
  * VARIABLES *
  * *****

40 LET menu=1000
50 LET Visualizacion=2000
100 REM

  *****
  * BUCLE CENTRAL *
  * *****

110 GO SUB menu
120 INPUT "Unidad a transformar >>> ",unidad
125 IF unidad<1 OR unidad>7 THEN
  N GO TO 120
130 CLS
140 INPUT "Cantidad >>> ",valor
150 IF valor<0 OR valor>9999999
  9 THEN GO TO 140
160 LET operation=1000+(unidad-
100)
170 GO SUB operation
180 GO SUB visualizacion
190 INPUT "Quiere continuar (S/
N) >>> LINE a$
200 IF a$="S" OR a$="s" THEN CL
  S GO TO 100
210 IF a$="N" OR a$="n" THEN ST
  OP
220 GO TO 190
999 REM

  *****
  * ARRITNRS *
  * *****

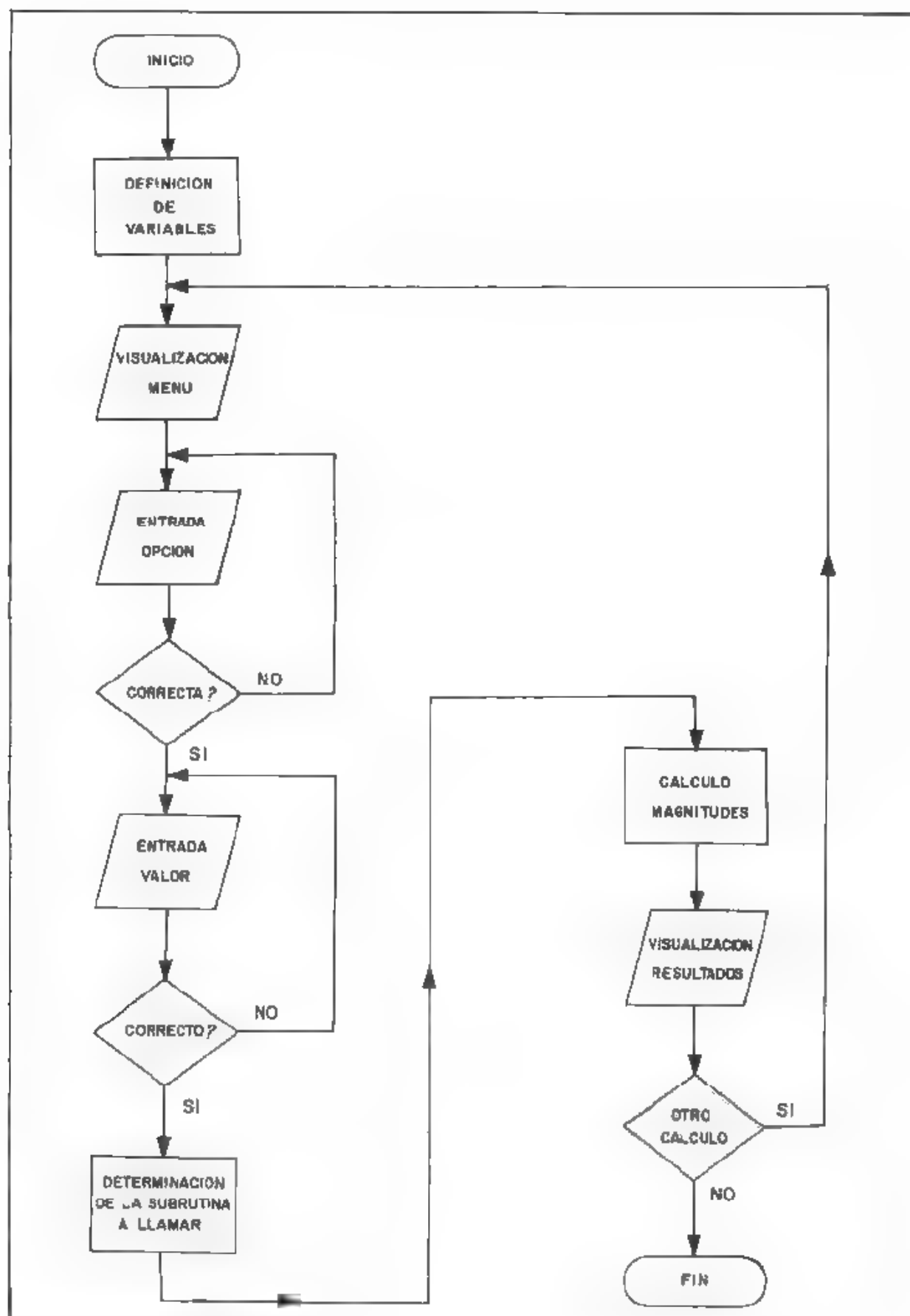
1000 REM MENU
1010 PRINT AT 1,5;"UNIDADES DE L
ONGITUD"
1020 PRINT AT 2,5,"-----"
1030 PRINT AT 6,8,"1 - Milimetro"
1040 PRINT AT 8,8,"2 - Centimetr
0."
1050 PRINT AT 10,8,"3 - Decimetr
0."
1060 PRINT AT 12,8,"4 - Metro."
1070 PRINT AT 14,8,"5 - Decametr
0."
1080 PRINT AT 16,8,"6 - Hectomet
ro."
1090 PRINT AT 18,8,"7 - Kilometr
0."
1095 RETURN
1100 REM MILIMETRO
1110 LET milimetro=valor
1120 LET centimetro=milimetro/10
1130 LET decimetro=centimetro/10
1140 LET metro=decimetro/10

```

```

1150 LET decanetro=metro/10
1160 LET hectometro=decanetro/10
1170 LET kilometro=hectometro/10
1180 RETURN
1200 REM CENTIMETRO
1210 LET centimetro=valor
1220 LET milimetro=centimetro*10
1230 LET decimetro=centimetro/10
1240 LET metro=decimetro/10
1250 LET decanetro=metro/10
1260 LET hectometro=decanetro/10
1270 LET kilometro=hectometro/10
1280 RETURN
1300 REM DECANETRO
1310 LET decimetro=valor
1320 LET milimetro=decimetro*100
1330 LET centimetro=decimetro*10
1340 LET metro=decimetro/10
1350 LET decanetro=metro/10
1360 LET hectometro=decanetro/10
1370 LET kilometro=hectometro/10
1380 RETURN
1400 REM METRO
1410 LET metro=valor
1420 LET milimetro=metro*1000
1430 LET centimetro=metro*100
1440 LET decimetro=metro/10
1450 LET decanetro=metro/10
1460 LET hectometro=decanetro/10
1470 LET kilometro=hectometro/10
1480 RETURN
1500 REM HECTOMETRO
1510 LET decanetro=valor
1520 LET milimetro=decanetro*100
  00
1530 LET centimetro=decanetro*10
  00
1540 LET decimetro=decanetro*100
1550 LET metro=decanetro*10
1560 LET hectometro=decanetro/10
1570 LET kilometro=hectometro/10
1580 RETURN
1600 REM KILOMETRO
1610 LET hectometro=valor
1620 LET milimetro=hectometro*10
  000
1630 LET centimetro=hectometro*1
  000
1640 LET decimetro=hectometro*10
  00
1650 LET metro=hectometro*100
1660 LET decanetro=hectometro*10
1670 LET kilometro=hectometro/10
1680 RETURN
1700 REM KILOMETRO
1710 LET kilometro=valor
1720 LET milimetro=kilometro*100
  000
1730 LET centimetro=kilometro*10
  000
1740 LET decimetro=kilometro*100
  00
1750 LET metro=kilometro*1000
1760 LET decanetro=kilometro*100
1770 LET hectometro=kilometro/10
1780 RETURN
2000 REM VISUALIZACION
2010 PRINT AT 2,10;"RESULTADOS:"
2020 PRINT AT 3,10,"-----"
2030 PRINT AT 8,1,"Milímetros ..
  ., milimetro
2040 PRINT AT 8,1,"Centímetros .
  ., centimetro
2050 PRINT AT 10,1,"Decímetros ,
  ., decanetro
2060 PRINT AT 12,1,"Metros .....
  ., metro
2070 PRINT AT 14,1,"Decametros .
  ., decanetro
2080 PRINT AT 16,1,"Hectometros
  ., hectometro
2090 PRINT AT 18,1,"Kilometros .
  ., kilometro
2100 RETURN

```



Estructura programa «longitud».

	subrutina «TEMPORIZACION».	la variable «reconocimiento»; ésta puede tener tres valores, «0» si el valor de «a\$» no corresponde con ninguna de las pistas (M, N o I), «1» si la opción elegida es «M» o «N», y «2» si la opción es «I».		za el equivalente en las restantes unidades.
1100-1130 :	Subrutina «ERROR». Visualiza un mensaje de error durante un tiempo. La subrutina «TEMPORIZACION» es utilizada.			Su estructura general es:
1200-1230 :	Subrutina «TEMPORIZACION». Tiene como parámetro de entrada la variable «retardo», dependiendo de este valor el tiempo de temporización. Aproximadamente se consigue un segundo de retardo, asignando el valor «100» a dicha variable.	1500-1590	Subrutina «ACIERTO». Se encarga de visualizar, cuando el ordenador acierta, el número de intentos realizados; al cabo de, aproximadamente, cuatro segundos visualiza el record de todas las jugadas.	10 : Comentario con el nombre del programa. 20 : Asignación de los colores verde para fondo y borde, y azul para los caracteres. 40-50 : Definición de variables. 110 : Llamada a la subrutina «MENU». 120 : Entrada «unidad» 125 : Verificación de la selección tiene que estar comprendida entre «1» y «7».
1300-1330 :	Subrutina «CALCULO». Calcula un número en función de los parámetros de entrada «mayor» y «menor», el resultado es devuelto al programa principal en el parámetro de salida «numero». La variable «igual» es también utilizada como parámetro de salida, si su valor es igual a «1» es que el ordenador está seguro del número.	1600-1650	Subrutina «SEGLIR». Comienza de nuevo el juego si se introduce la letra «S» (SI) o se termina si es la «N» (NO). Si se pulsa otra letra, se ejecuta una llamada a la subrutina «ERROR».	130 : Borrado pantalla 140 : Entrada «valor». 150 : Verificación de la magnitud, debe estar comprendida entre «0» y «99999999».
1400-1470 :	Subrutina «VERIFICACION». Tiene como parámetro de entrada la variable «a\$», en función de su valor, modifica e de las variables «mayor» y «menor». El parámetro de salida está asignado en		El programa número «2», realiza transformaciones entre magnitudes expresadas en unidades de longitud.	160 : Cálculo del número de línea, correspondiente a la subrutina encargada de realizar las transformaciones elegidas. 170 : Llamada a la subrutina calculada en la línea 160. 180 : Llamada a la subrutina de visualización de resultados.
			Presenta un menú con las diversas unidades; para elegir una de ellas, es necesario introducir el código correspondiente y, a continuación, la magnitud. Una vez realizados los cálculos, el programa visuali-	1900-220 : Rutina para saltar al principio y realizar otro cálculo o, por el contrario, terminar
				1000-1095 Subrutina «MENU». Visualiza el menú con las diversas opciones.
				1100-1180 Subrutina «MILI

	METRO». Asigna a la variable «milímetro» el contenido de la variable «valor». A partir de este dato realiza las equivalencias con las restantes unidades.				
1200-1280	Subrutina «CENTI-METRO». Idem con la variable «centímetro»	1300-1380	Subrutina «DECI-METRO». Idem con la variable «decímetro».	1700-1780	Subrutina «KILO-METRO». Idem con la variable «kilómetro»
		1400-1480	Subrutina «METRO». Idem con la variable «metro».	2000-2100	Subrutina «VISUALIZACION». Visualiza en pantalla los resultados obtenidos con cualquiera de las anteriores subrutinas. ■
		1500-1580	Subrutina «DECA-METRO». Idem con la variable «decámetro».		
		1600-1680	Subrutina «HECTOMETRO». Idem con la variable «hectómetro».		

DATOS DE UN PROGRAMA

Cuando estudiamos las sentencias «LET» e «INPUT», vimos que se podían asignar o introducir datos en nuestros programas; pero cuando estos datos son de valor constante y numerosos, podemos utilizar las sentencias «DATA» para almacenarlos y «READ» para leerlos.

READ

Acceso al teclado



MODOS E

Tipo de sentencia

Comando de entrada.

Definición

Con esta sentencia, se pueden leer los datos que haya dentro de un programa si su estructura es

SENTENCIA	ARGUMENTO
READ	variable, variable...

Las variables pueden ser tanto numéricas como de cadena, pero teniendo en cuenta que deben ir en concordancia con el dato leído, y separadas por comas.

Ejemplos:

— READ día, mes.

- READ J\$.
- READ numero, a\$, nota.
- READ valor

El dato es asignado a la variable del argumento.

DATA

Acceso al teclado



MODOS E

Tipo de sentencia

Datos.

Definición

Los valores que forman el argumento de esta sentencia, sirven para ser asignados a las variables de las instrucciones «READ», y forman, por tanto, tablas de datos.

La estructura de esta sentencia es:

SENTENCIA	ARGUMENTO
DATA	constante, constante

Las constantes pueden ser numéricas o alfanuméricas.

Ejemplos:

- DATA 30, 47, 3225, 17 14
- DATA «MICROHOBBY», «CURSO», «BASIC»
- DATA «SPECTRUM», 23, 10, «MICRO»
- DATA 8

Los valores del argumento deben ir separados por comas y pueden estar incluidos en cualquier parte del programa, incluso después de una sentencia «STOP», ya que el ordenador los ignora mientras no se ejecute una instrucción «READ»

Ejemplo.

```
50 DATA 10 20 67
100 DATA 5, 15, 27, 33
170 DATA «MICRO», «CHIP»
300 DATA «ENTRADA», 30
```

aunque por razones de claridad y estructuración del programa, conviene que estén todos juntos al principio o a final de éste.

Ejemplo:

```
1000 DATA 10 20 67,5
1010 DATA 15 27 33
1020 DATA «MICRO», «CHIP», «ENTRADA»
1030 DATA 30
```

La cantidad de valores a incluir dentro de una instrucción «DATA», viene determinada únicamente por la capacidad de una línea (22 filas)

Utilización de «READ» y «DATA»

Cuando el intérprete BASIC encuentra una sentencia de tipo

READ variable

analiza una por una todas las

líneas del programa, hasta que encuentra la primera que contiene «DATA», posteriormente, asigna a la variable el primer dato de ésta. La siguiente vez que encuentra «READ» se le asigna el segundo dato, y así sucesivamente.

Ejecute el siguiente programa:

```

10 REM *****
11 READ/DATA
12 *****
20 READ a$,b$
30 READ numero
40 READ f$
50 READ q$
60 PRINT "a$=";a$
70 PRINT "b$=";b$
80 PRINT "f$=";f$
90 PRINT "q$=";q$
100 REM *****
110 DATA "MICROHOBBY", "SEMANAL"
120 DATA 1985, "BASIC/SINCLAIR"
130 DATA "SPECTRUM", 16

```

según se van ejecutando las sentencias «READ», se van asignando los sucesivos datos, así...

a\$ tendrá el valor «MICROHOBBY»

b\$ = «SEMANAL»

número = 1985

f\$ = «BASIC/SINCLAIR»

q\$ = «SPECTRUM»

valor = 16

Para conocer el ordenador cuál es el siguiente dato que tiene que leer, la tabla de datos tiene un *puntero* o *índice* que lo señala y que se incrementa en uno cada vez que se hace una lectura.

Una cosa importante es que no pueden realizarse más lecturas que datos haya en el programa, ya que tendríamos un mensaje de error; sin embargo, puede haber más datos que lecturas.

La instrucción «READ» puede formar parte de un bucle,

de esta manera puede agilizar-se la lectura de datos.

Ejemplo:

```

10 REM *****
11 BUCLE READ
12 *****
15 LET total=0
20 FOR n=1 TO 22
30 READ numero
40 LET total=total+numero
50 PRINT numero,total
60 NEXT n
70 REM *****
80 DATA 23,45,57 48 51,57,59,3
90 DATA 12,45 83,27,54 12,67 7
100 DATA 34,55,71 89,21 83

```

Dentro de una tabla de datos puede leerse uno determinado; en la siguiente aplicación, introduciendo un número entre 1 y 12, el ordenador nos visualiza el mes correspondiente.

```

10 REM *****
11 MESES
12 *****
20 INPUT "Mes (1 12) :"; mes
30 IF mes < 1 OR mes > 12 THEN GO TO 20
40 LET puntero=1
50 READ mes
60 IF puntero > mes THEN LET puntero=puntero+1
70 PRINT puntero
80 REM *****
90 DATA "ENERO", "FEBRERO", "MARZO", "ABRIL", "MAYO", "JUNIO", "JULIO", "AGOSTO", "SEPTIEMBRE", "OCTUBRE", "NOVIEMBRE", "DICIEMBRE"

```

Veamos otro ejemplo de aplicación de las sentencias «READ/DATA».

```

10 REM *****
11 PRIMOS
12 *****
15 PRINT "Número primo = Cuadrado"
20

```

```

17 PRINT " "
20 READ p, q
30 IF NOT primo THEN STOP
40 LET cladr=q*q
50 PRINT p, q, cladr
60 GO TO 20
180 REM *****
190 DATA 1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53

```

En este caso, el ordenador calcula el cuadrado de los números primos menores de cincuenta, que se han introducido como constantes en una serie de «DATAS». Al final de los datos se introduce el valor «0», como código de ruptura; cuando se realiza la lectura de este código, el programa se para, de esta manera no es necesario llevar la cuenta de la cantidad de datos. Para preguntar por esta condición se ha utilizado:

IF NOT primo THEN STOP

la condición se hace verdadera y por tanto se ejecuta «STOP», cuando la variable «primo» es igual a «0», ver función «NOT» (pág. 35).

RESTORE

Acceso al teclado



Tipo de sentencia

Comando de programación.

Definición

Una de las ventajas de las tablas de datos es poder leer éstos cuantas veces deseemos. Para volver al principio de la tabla o a cualquier parte de ella, se necesita restaurar el puntero o índice, esto se consigue con la sentencia «RESTORE».

Su estructura es la siguiente:

SENTENCIA	ARGUMENTO
RESTORE	N.º de línea

Ejemplos:

- RESTORE 40.
- RESTORE 1230.
- RESTORE.

Cuando el argumento se omite, el intérprete BASIC toma por defecto el valor 0. Al ejecutar un programa con el comando directo «RUN» inicia el puntero en la primera sentencia «DATA».

Si no se restaura el puntero al finalizar la lectura de la tabla, el siguiente dato que se quiera leer provocará un mensaje de error.

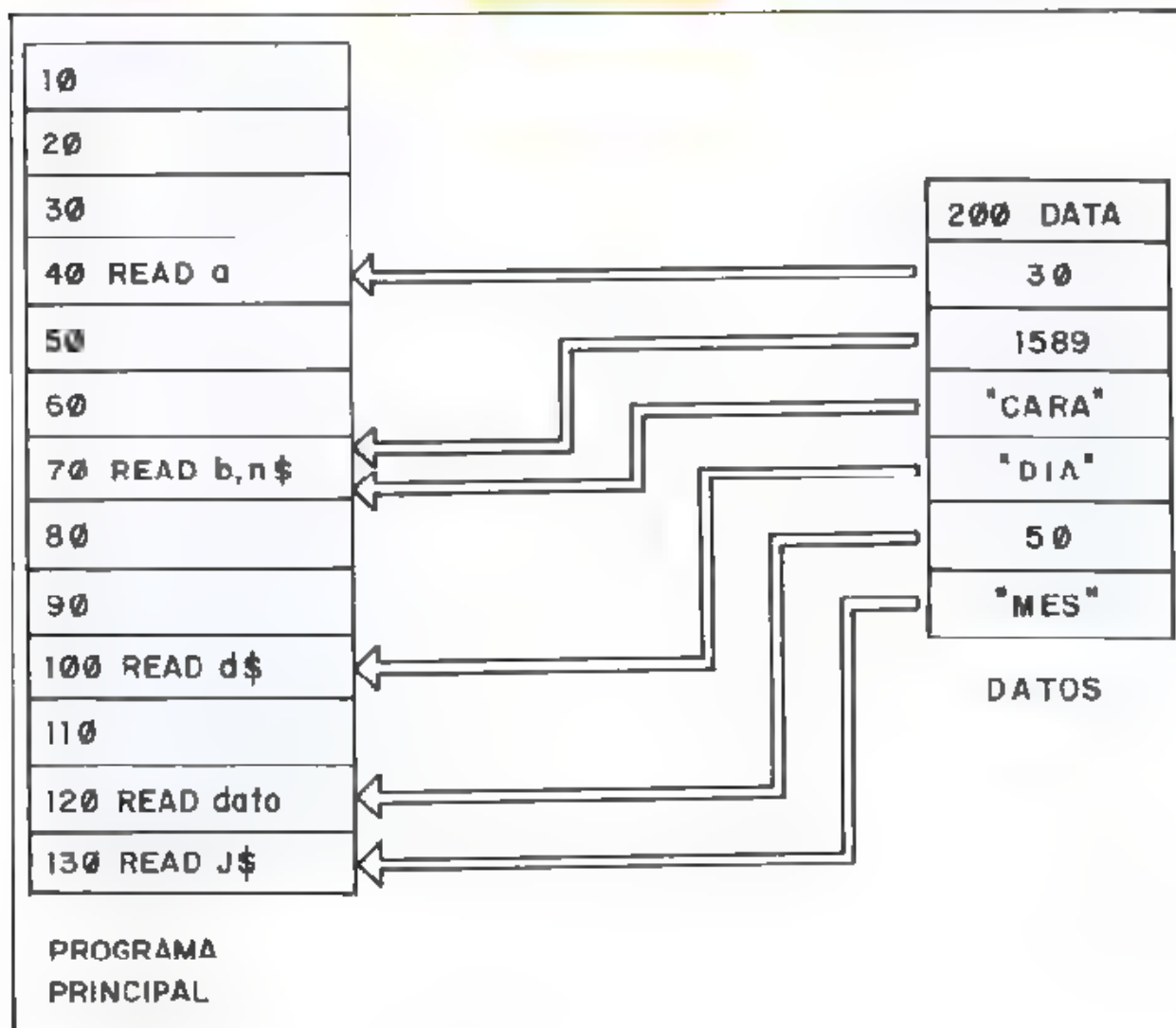
Ejemplo:

```
10 READ código
20 READ precio
30 PRINT código
40 PRINT precio
50 RESTORE
80 READ número
```

```
70 READ valor
80 PRINT número
90 PRINT valor
100 DATA 27, 3500
```

La sentencia «RESTORE» de la línea 50, permite leer la tabla dos veces. En la primera lectura, el valor «27» es asignado a la variable «código»; sin embargo, en la segunda se asigna a la variable «número»; otro tanto ocurre con el valor «3500» que se asigna primero a la variable «precio», y posteriormente a «valor».

En el siguiente programa tiene como argumento la sentencia «RESTORE», una variable de tipo numérico, cuyo va-



Lectura de datos.

lor está en función de un «INPUT»; dependiendo de éste la inicialización del puntero y, por tanto, del acceso a los datos.

```

10 REM *****
11 RESTORE
12 *****
20 INPUT "Clave 1 a 5: "; C
30 IF C < 1 OR C > 5 THEN
40 GOTO 20
50 LET index=1000+C*(10)
60 RESTORE index
70 READ A$ TO A
80 PRINT A$
90 NEXT C
1000 REM *****
1010 DATOS *****
1020 DATA 1 2 3 4
1030 DATA 5 6 7 8
1040 DATA 9 10 11 12
1050 DATA 13 14 15 16
1060 DATA 17 18 19 20

```

Errores

Cuando se manejan sentencias del tipo READ/DATA, hay tres tipos de error que suelen producirse frecuentemente:

- Cuando se ejecuta una sentencia «READ» y el puntero se encuentra al final de la tabla, indicando que no hay más datos. El mensaje que se visualiza es:

E Out of DATA

Ejemplo.

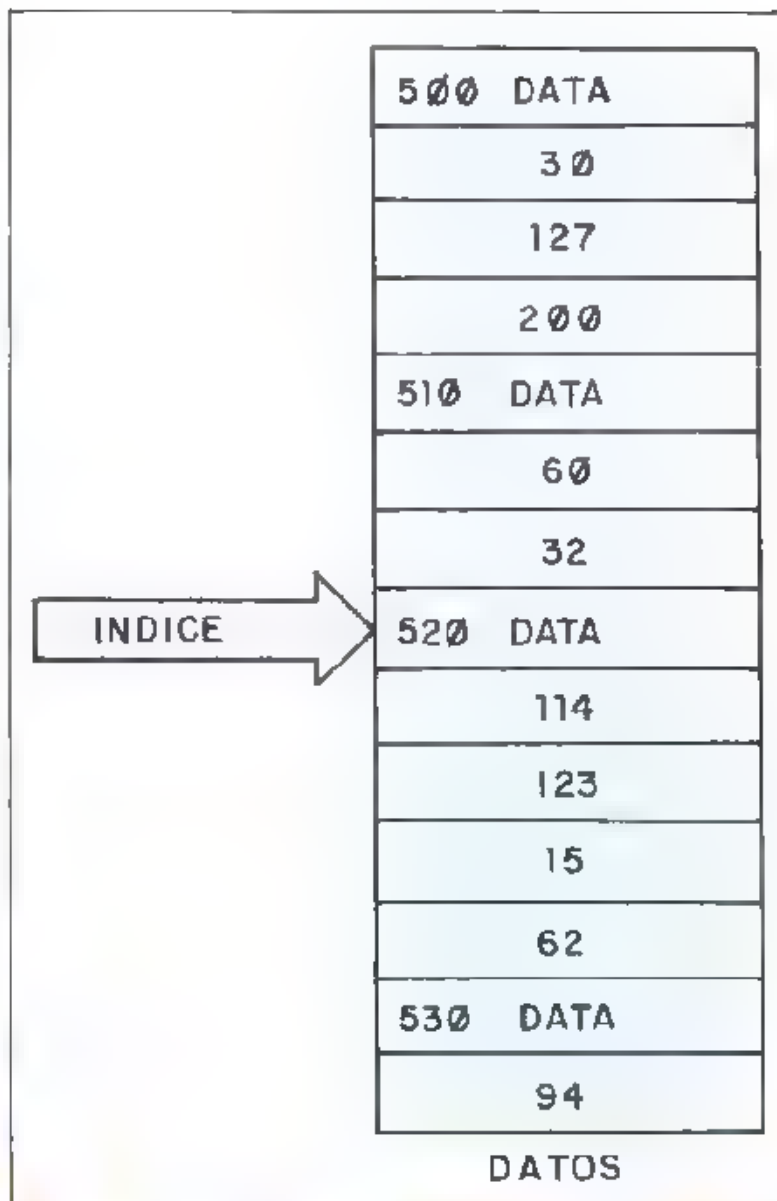
```

10 REM *****
11 ERROR 1 *****
20 FOR I=1 TO 5
30 READ A$
40 PRINT A$
50 NEXT I
100 REM *****
1010 DATOS *****
110 DATA 10 23 55,78

```

se pretende leer cinco datos, cuando en realidad sólo hay cuatro.

- Cuando al confeccionar la tabla de datos se incluye un valor numérico en lugar de uno alfanumérico, o viceversa. Al ejecutarse di-



Mecanismo «Restore 520».

cho programa y tratar de leer un dato que no está en concordancia con el de la variable que acompaña a «READ», se produce el error:

C Nonsense in BASIC

Ejemplo:

```

10 REM *****
11 ERROR 2 *****

```

```

10 PRINT "NOTA ALUMNO"
20 PRINT
30 FOR I=1 TO 4
40 READ A$ TO A
50 PRINT A$
60 NEXT I
100 REM *****
1010 DATOS *****
110 DATA 5 ANTONIO 5 GERARDO
120 DATA 2 SEBASTIAN 10 LUIS
130 DATA 7 CARLOS 4 PABLO
140 DATA 3 4 5

```

En la línea 140, el segundo dato debería ser alfanumérico.

PROGRAMA 1

```

10 REM *****
   *      *
   *  CURSO/BASIC  *
   *      *
   * *****      *
   *  GEOGRAFIA 1  *
   *      *
   * *****      *

20 BORDER 4. PAPER 4 INK 1: C
L
30 GO SUB 1000
40 REM *****
   *      *
   *  BUCLE PRINCIPAL  *
   *      *
   * *****      *

50 PRINT AT 1,3,"AUTONOMIAS DE LA PENINSULA"
   PRINT AT 2,3,""

70 FOR c=5 TO 19
80 READ a$
90 LET opcion c-4
100 PRINT AT c,3,opcion
110 IF opcion<10 THEN PRINT "

120 PRINT " - ,a$
130 NEXT c
140 INPUT AT 0 1, "Que Autonomia desea visualizar" >>> "opcion
150 IF opcion<1 OR opcion>15 THEN GO TO 140
160 CLS
170 LET direccion 2000+10opcion+
100
180 RESTORE direccion
190 READ a$
200 PRINT AT 1,3,a$
210 PRINT "

220 READ numero
230 FOR n=1 TO numero
240 PRINT "
250 READ a$,b$
260 PRINT a$,"l",b$,"
270 NEXT n
280 INPUT "Pulse 'ENTER' para retornar a LINE a$
290 RESTORE 1100
300 CLS
310 GO TO 40
1000 REM *****
   *      *
   *  DEFINICION 'N'  *
   *      *
   * *****      *

1010 RESTORE 1060
1020 FOR d=USR "n" TO USR "n"+7
1030 READ dato
1040 POKE d,dato
1050 NEXT d
1060 DATA 24,66,98,82,74,70,66,0
1070 RETURN
1100 REM *****
   *      *
   *  MENU  *
   *      *
   * *****      *

1110 DATA "ANDALUCIA", "ARAGON", "ASTURIAS", "CANTABRIA", "CASTILLA-LA MANCHA",
1120 DATA "CASTILLA LEON", "CATALUÑA", "COMUNIDAD VALENCIANA", "EXTREMADURA", "GALICIA",
1130 DATA "MADRID", "MURCIA", "NAV

```

```

ARRA", "PAIS VASCO", "LA RIOJA"
2000 REM *****
   *      *
   *  PROVINCIAS  *
   *      *
   * *****      *

2100 DATA "ANDALUCIA", 0, "ALMERIA", "ALMERIA", "CADIZ", "CADIZ", "CORDOBA", "CORDOBA", "GRANADA", "GRANADA", "HUELVA", "HUELVA", "JAEN", "JAEN", "MALAGA", "MALAGA", "SEVILLA", "SEVILLA"
2200 DATA "ARAGON", 3, "HUESCA", "HUESCA", "TERUEL", "TERUEL", "ZARAGOZA", "ZARAGOZA"
2300 DATA "ASTURIAS", 1, "ASTURIAS", "OVIEDO"
2400 DATA "CANTABRIA", 1, "CANTABRIA", "SANTANDER"
2500 DATA "CASTILLA-LA MANCHA", 5, "ALBACETE", "ALBACETE", "CIUDAD REAL", "CIUDAD REAL", "CUENCA", "CUENCA", "GUADALAJARA", "GUADALAJARA", "TOLEDO", "TOLEDO"
2600 DATA "CASTILLA-LEON", 9, "AVILA", "AVILA", "BURGOS", "BURGOS", "LEON", "LEON", "PALENCIA", "PALENCIA", "SALAMANCA", "SALAMANCA", "SEGOVIA", "SEGOVIA", "SORIA", "SORIA", "VALLADOLID", "VALLADOLID", "ZAMORA", "ZAMORA"
2700 DATA "CATALUÑA", 4, "BARCELONA", "BARCELONA", "GERONA", "GERONA", "LERIDA", "LERIDA", "TARRAGONA", "TARRAGONA"
2800 DATA "COMUNIDAD VALENCIANA", 3, "ALICANTE", "ALICANTE", "CASTELLON", "CASTELLON", "C. DE LA PLANA", "C. DE LA PLANA", "VALENCIA", "VALENCIA"
2900 DATA "EXTREMADURA", 2, "BADAJOZ", "BADAJOZ", "CACERES", "CACERES"
3000 DATA "GALICIA", 4, "LA CORUÑA", "LA CORUÑA", "LUGO", "LUGO", "ORENSE", "ORENSE", "PONTEVEDRA", "PONTEVEDRA"
3100 DATA "MADRID", 1, "MADRID", "MADRID"
3200 DATA "MURCIA", 1, "MURCIA", "MURCIA"
3300 DATA "NAVARRA", 1, "NAVARRA", "PAMPLONA"
3400 DATA "PAIS VASCO", 3, "ALAVA", "ALAVA", "GUIPUZCOA", "GUIPUZCOA", "SAN SEBASTIAN", "SAN SEBASTIAN", "VIZCAYA", "VIZCAYA", "BILBAO", "BILBAO"
3500 DATA "LA RIOJA", 1, "LA RIOJA", "LOGRONO"

2270 DATA "SELLA", "VALLE SAJAMBRA", "LEON", 1, "PONGA"
2300 DATA "MEDITERRANEA"
2310 DATA "EBRO", "FONTIBRE", "SANTANDER", "ARAGON", "EGRA", "GALLEGO", "GUADALOPE", "HUERA", "JALON", "SEGRE"
2320 DATA "GUADALHORCE", "SIERRA DE GIBALTO", "MALAGA GRANADA", 0
2330 DATA "JUCAR", "CERRO SAN FELIPE", "CUENCA", 1, "CABRIE"
2340 DATA "LOBREGAT", "FONT DEL LOBREGAT", "BARCELONA", 2, "CARONER", "NOYA"
2350 DATA "SEGURA", "SIERRA DEL SEGURA", "JAEN", 3, "ARGOS", "BENAMOR", "MUNDO"
2360 DATA "TER", "MONT COSTA BOND", "GERONA", 2, "FRESSER", "ONAR"
2370 DATA "TURIA", "HUELA SAN JUAN", "TERUEL", 0

```

c) En las tablas de datos con valores de cadena, puede suceder que se nos o y de

colocar las correspondientes comillas. El intérprete BASIC, al ejecutarse el pro-

grama, tomará dicho dato como variable numérica, si no existe ninguna variable

Ejemplo:

2 Variable not found

si por el contrario existiera, nos encontraríamos en el caso expuesto en el apartado «b».

```

30 REM *****
    1
    2 ERROR J
    3
    4 *****

```

```

20 FOR N=1 TO 9
30 READ A$
40 PRINT A$
50 NEXT A
60 REM

```

```

      B7EED+010
      F DRTD5 F
      F FFF+00B
      F FFF+00C

```

128 DATA AUTOMOBILE BICYCLE
AUTOS HELICOPTER
128 DATA MOTO BARO AVION
UNDERGROUND

En la línea 120, el dato BARCO debería ir entre comillas.

PROGRAMA 2

```

10 REM
   *
   * CURSOS/BASICO
   *
   * *****
   * GEOGRAFIA 2
   *
   * *****

20 BORDER 1; PAPER 4: INK 0: C
LS
30 GO SUB 1000
40 REM *****
   *
   * BUCLE PRINCIPAL
   *
   * *****

50 PRINT AT 1,9;"RIOS ESPAÑOLE
60 PRINT "_____
70 PRINT AT 5,8 "_____
80 PRINT AT 7,11,"VERTIENTES"
90 PRINT AT 8,8,"
100 PRINT AT 12,0, 1 - ATLANTIC
110 PRINT AT 15,8;"2 - CANTABRI
120 PRINT AT 18,8;"3 - MEDITERR
130 INPUT "Que vertiente desea
>>> vertiente
140 IF vertiente<1 OR vertiente
>3 THEN GO SUB 1100 GO TO 110
150 LET rios=2000+(vertientes-1
0)
160 RESTORE rios
170 READ numero,*
180 GO SUB 1200
190 INPUT "Que rio desea >>> ";
rio
200 IF rio<1 OR rio>numprio THEN
GO SUB 1100. GO TO 170
210 LET direccion=n-(numero-ri
o+1)*10)
220 RESTORE direccion
230 CLS
240 GO SUB 1300
250 INPUT "Pulse 'ENTER'" para
retornar ", LINE 2$
260 CLS
270 GO TO 50
1000 REM *****
   *
   * DEFINICION "N"
   *
   * *****

1010 RESTORE 1060
1020 FOR d=USR 'n' TO USR 'n'+7
1030 READ dato
1040 POKE d,dato
1050 NEXT d
1060 DATA 24,56,98,82,74,70,66,0
1070 RETURN
1100 REM *****
   *
   * ERROR
   *
   * *****

```

[illegible]

```

CIN , TERUEL , 7 , ALAGON , ALBERJ
HE , ALMONTE , "GUADARRAMA" , GUAD
IELA , "JARAMA" , TIETAR
2170 DATA TABORE , MONTES DE BO
DELO , LA CORUNA , 2 , SAMO , LENG
UELE ,
2180 DATA ULLA , MONTE CEBREIRO
LUGO , 3 , VEZ , ISU , SHK
2200 DATA , CANTABRICA
2210 DATA "BIDASOA" , HAYA , NAVA
ARR , 0

```

```

2220 DATA "EO" , MONTES DEL CADAB
O , LUGO , 0
2230 DATA "NALON" , "PUERTO Tarna"
, OJJEDO , 4 , "CALDAL" , "NARCEA" , "N
ORA" , "TRUBIA"
2240 DATA NAVIA , SIERRA RAMADO
IRO , LUGO , 0
2250 DATA "SERVION" , "PEÑA DE ORD
UNA" , "ALBUA" , 1 , IBAIZABAL
2260 DATA "ORIA" , MONTE AITZGORA
I" , GLIPJZCOA , 0

```

Programas

Como aplicación de las sentencias «READ/DATA», se presentan, en esta ocasión, dos programas de utilidad didáctica. Ambos están relacionados con un área muy importante dentro de la enseñanza: la Geografía.

El alfabeto español tiene una letra que no está incluida en el juego de caracteres del Spectrum, ésta es la «Ñ»; por tanto ha sido necesario diseñar un nuevo gráfico (GDU) con dicha forma. La técnica empleada para realizarlo será explicada en el capítulo dedicado a gráficos. Para acceder a la «Ñ» es necesario pasar a modo gráfico (**G**), como recordará, basta con pulsar simultáneamente la tecla «CAPS SHIFT» y «9» una vez en este modo, debe pulsarse la tecla «N»

ATENCIÓN

Hasta que el programa no se ejecute, no se visualizará el gráfico correspondiente a la «Ñ», por tanto no se preocupe si ésta no aparece al pasar al modo **G**.

Para retornar al modo anterior pulse la tecla «9».

El primer programa está dedicado al estudio de las *Autonomías*. Al ejecutar el programa, se presenta un menú con las distintas Autonomías de la Península. Cada una de ellas tiene a su izquierda un número de opción, éste sirve como

referencia para seleccionarla. Una vez elegida la opción, aparece en la parte superior de la pantalla el nombre de la Autonomía, y a continuación, el nombre de las provincias que la componen con sus correspondientes capitales, encerradas entre paréntesis.

La estructura del programa es

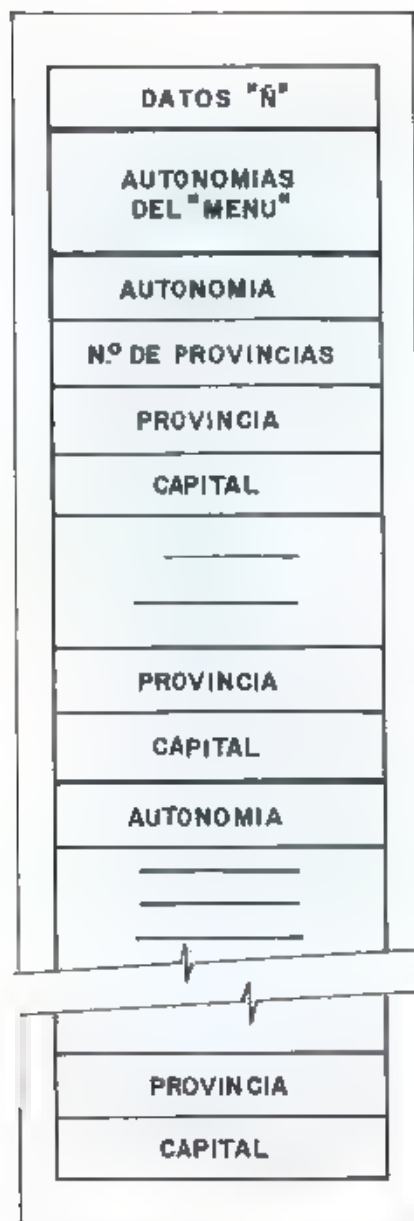
- 10 : Comentario con el nombre del programa.
- 20 : Asignación del color verde para fondo y papel y azul para los caracteres.
- 30 : Llamada a la subrutina que define como GDU, la «Ñ»
- 40 : Comienzo del programa principal
- 50-60 : Rótulo de presentación
- 70-130 : Bucle para visualizar el número de opción y el nombre de las Autonomías, éstas están contenidas en una tabla de datos.
- 140-150 : Entrada opción y comprobación.
- 160 : Borrado pantalla.
- 170 : Algoritmo empleado para el cálculo del número de línea, donde debe iniciarse el puntero de la tabla de datos.

- 180 : Inicialización del puntero (RESTORE).
- 190-210 : Lectura y visualización del nombre de la Autonomía.
- 220 : Lectura del número de provincias.
- 230-270 : Bucle para la lectura y visualización de las provincias y sus capitales.
- 280 : Entrada de la tecla «ENTER»
- 290 : Inicialización de la tabla.
- 300-310 : Borrado pantalla y regreso al menú principal
- 1000-1070 : Subrutina para definición de la tecla «Ñ»
- 1100-1130 : Datos del menú principal.
- 2000-3500 : Datos correspondientes a las provincias y sus capitales.

El segundo programa sirve para estudiar o consultar sobre la hidrografía española. Al ejecutarse presenta en pantalla el menú con las tres vertientes:

- ATLANTICA
- CANTABRICA.
- MEDITERRANEA

Al elegirse una de ellas aparece otro menú con los ríos que desembocan. Al seleccionar uno de ellos, se visualiza



Estructura de datos del programa "Autonomías".

en pantalla la información correspondiente a.

- Nombre del río.
- Nacimiento.
- Provincia.
- Afluentes más importantes (si los tiene).

Pulsando «ENTER» se retorna al menú principal. Si al elegir las opciones se selecciona una que no existe, aparece en la parte inferior de la pantalla un mensaje de error.

La estructura del programa es la siguiente:

- | | | |
|---------|---|---|
| 10 | : | Comentario con el nombre del programa. |
| 20 | : | Asignación del color azul para el borde, verde para el fondo y negro para los caracteres. |
| 30 | : | Llamada a la subrutina que define la «N». |
| 40 | : | Comienzo del programa principal. |
| 50-100 | : | Presentación del menú con las vertientes. |
| 110-120 | : | Entrada de opción y comprobación. |
| 130 | : | Algoritmo que calcula el número de línea donde debe inicializarse la lectura de los ríos. |
| 140 | : | Inicialización del puntero. |
| 150 | : | Leer número de ríos y vertiente. |
| 160 | : | Llamada a la subrutina que lee y visualiza el menú con los nombres de los ríos. |
| 170-180 | : | Entrada de la opción «río» y verificación. |
| 190 | : | Algoritmo para calcular dónde debe inicializarse la lectura de los datos correspondientes al río elegido. |
| 200 | : | Inicialización del puntero. |
| 210 | : | Borrado pantalla. |
| 220 | : | Llamada a la subrutina que lee y visualiza los datos correspondientes al dato elegido. |



Estructura de datos del programa "Ríos".

- | | | |
|-----------|---|--|
| 230 | : | Entrada de la tecla «ENTER». |
| 240-250 | : | Borrado pantalla y vuelta al menú principal (vertientes). |
| 1000-1070 | : | Subrutina para definir la letra «N». |
| 1100-1130 | : | Subrutina de presentación del mensaje «ERROR». |
| 1200-1290 | : | Subrutina para leer y visualizar los ríos. |
| 1300-1410 | : | Subrutina para leer y visualizar los datos, (nacimiento, provincia, ...) |
| 2000-2370 | : | Tabla de datos. ■ |

LECTURA DEL TECLADO Y TEMPORIZACIONES

INKEYS

Acceso al tedado

IN KEY \$



MOD0 **E**

OVER

Tipo de sentencia

Función de entrada.

Definition

La función «INKEY\$» permite leer un solo carácter introducido por teclado. Se diferencia básicamente de la sentencia «INPUT» en,

- a) **INKEY\$** no produce eco es decir, que no se visualiza el valor de la tecla pulsada.
- b) El dato tiene que estar preparado al ejecutarse la función «**INKEY\$**», es decir, que no espera, como en el caso de «**INPUT**», a que se teclee el dato: por tanto, si no está preparado se ejecuta la siguiente instrucción y la función retorna una cadena vacía (" ").

«INKEY\$» no tiene ningún argumento, por el contrario, forma parte de los argumentos de sentencias del tipo:

- LET.
- PRINT.
- IF... THEN...

Ejemplos:

```

10 REM      A B C D E F G H I J
           0 1 2 3 4 5 6 7 8 9
           0 1 2 3 4 5 6 7 8 9
           0 1 2 3 4 5 6 7 8 9
           0 1 2 3 4 5 6 7 8 9
20 IF INKEY#="" THEN GO TO 20
30 PRINT INKEY#
40 GO TO 10

```

- 1) En el siguiente programa se visualiza el valor retornado por la función «INKEY\$» cuando se pulsa una tecla, es decir, cuando no es una cadena vacía.
observe que «INKEY\$» diferencia entre mayúsculas (modo **C**), minúsculas (modo **L**) o aquellos símbolos que se pulsan conjuntamente con SYMBOL SHIFT.

```

10 REM *****
      :
      : INKEY$ 2 4
      :
      : *****
20 INPUT "NUMBER 9 IS NUMBER 0"; N
30 PRINT "*****"
40 PRINT "GO AT 10, QUOTE 10"
50 IF INKEY$ = " " THEN GO TO 50
60 IF INKEY$ = " " OR INKEY$ = " "
THEN GO TO 20
70 IF INKEY$ = " " OR INKEY$ = " "
THEN STOP
80 GO TO 40

```

- 2) En este caso, «INKEY\$» se utiliza para preguntar sobre una tecla determinada. Tiene la ventaja sobre la utilización de «INPUT», en que no es necesario pulsar «ENTER».

```

10 REM 1111111111111111
   11
   12 14KEY$ 3
   13
   1411111111111111

```

```

20 PRINT AT 2 0 MENU PRINCIPA
L
30 PRINT AT 2 0
40 PRINT AT 5 10 1 - EDITAR
50 PRINT AT 13 10 2 - TITULO
60 PRINT AT 14 10 3 - BOBAAA
70 PRINT AT 14 10 4 - AR-RI-IA
80 PRINT AT 16 10 5 - INSERTA
90 PRINT AT 18 10 6 - FIM
100 LET A=Z:GOTO 100
110 IF A=1 THEN GOTO 1000
120 IF A=2 THEN GOTO 2000
130 IF A=3 THEN GOTO 3000
140 IF A=4 THEN GOTO 4000
150 IF A=5 THEN GOTO 5000
160 GOTO 100

```

- 3) Cuando el valor retornado por «INKEY\$», se utiliza en sentencias posteriores a la pulsación de la tecla, conviene asignar este valor a una variable de cadena.

PAUSE

Acceso al tedado

pi



MODOK

INVERSE

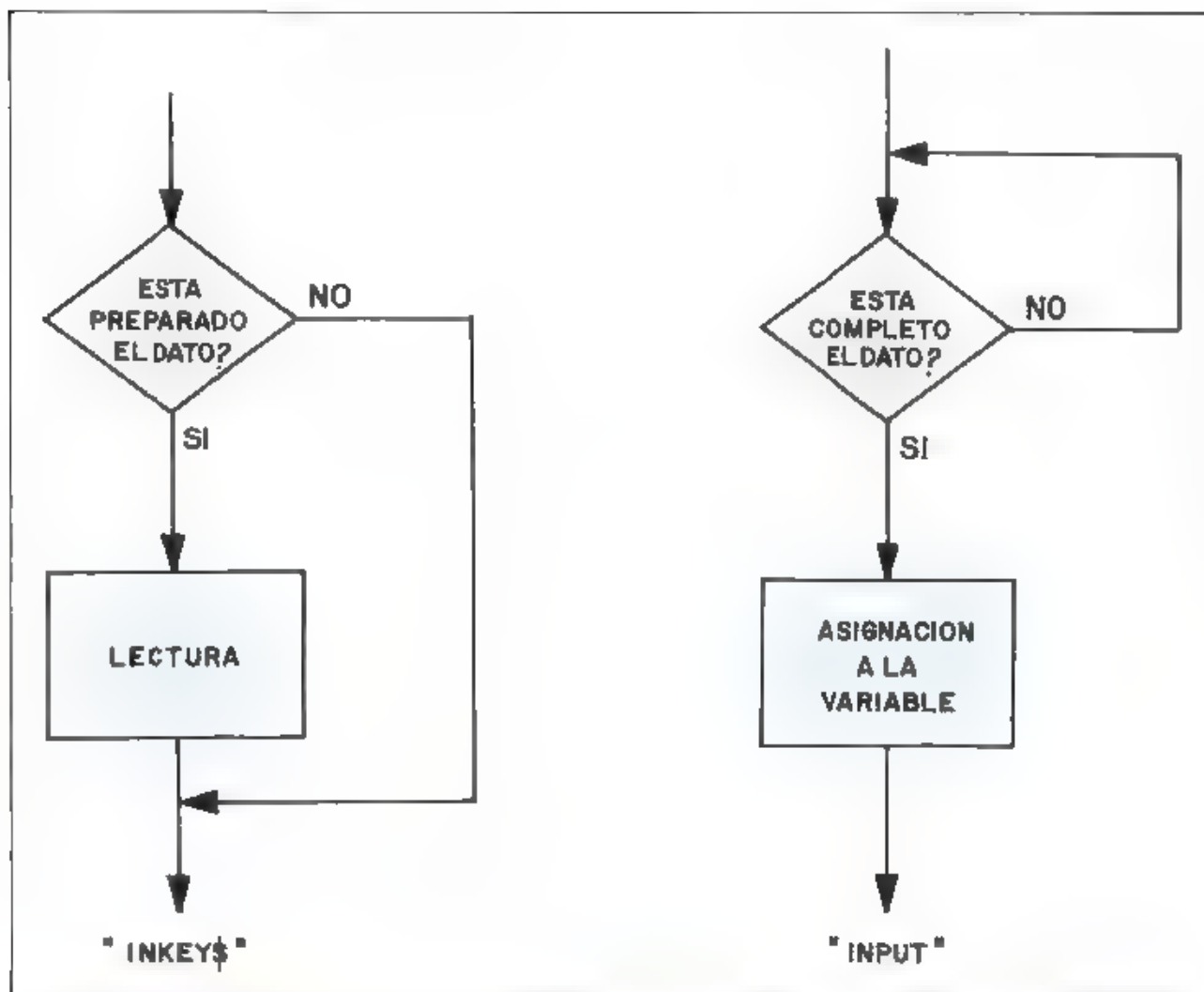
Tipo de sentència

Comando de programación.

Definición

Esta sentencia suspende la ejecución de un programa durante un tiempo determinado, es decir, *temporiza* o hace una *pausa*.

El efecto que produce es similar al proporcionado por las sentencias:



Diferencias entre "INKEY\$" e "INPUT".

```

10 FOR x = 1 TO tiempo
20 NEXT x

```

La estructura de este comando es:

SENTENCIA	ARGUMENTO
PAUSE	expresión numérica

Ejemplos:

- PAUSE 10.
- PAUSE 950.
- PAUSE 30 * 5.

Si se introduce un número fraccionario, éste se redondea al valor entero más cercano. El rango de valores está comprendido entre 0 y 65535, cuando se introduce uno fuera de margen aparece el mensaje.

B Integer out of range

La realización que existe entre el número del argumento y el tiempo de temporización, en segundos es:

$$tiempo = \frac{n}{50}$$

por lo tanto «PAUSE 100», detiene la ejecución del programa durante «2» segundos. Dividiendo entre 50 el valor máximo (65535) obtenemos el mayor tiempo de temporización:

$$65535/50 = 1310.7 \text{ sg.}$$

21 min. 50.7 sg.

ATENCIÓN

Si durante la ejecución de una sentencia «PAUSE» se presiona una tecla, la temporización se interrumpe y continúa el programa en la siguiente instrucción.

El valor «0», asignado a «PAUSE», provoca una temporización indefinida hasta que se pulsa una tecla.

Puede utilizarse una variable del tipo numérico como argumento de esta sentencia.

Ejemplo:

```

10 REM *****
   PAUSE VARIABLE *****

```

```

15 BORDER 4
20 INPUT TEMPORIZACION 1-10
   TEMP
30 IF TEMP 1 ON TEMP 10 TH
EN GO TO 20
40 LET PULSE=1, tempo*50
50 BORDER
60 PAUSE PULSE
70 BORDER 4
80 PRINT NO AT 1 0 Quiere con
100 PAUSE 0
110 LET AS=INKEY$
120 IF AS=" " OR AS=" " THEN GO
TO 20
130 IF AS=" " OR AS=" " THEN ST
OP
130 GO TO 90

```

El valor asignado a la variable «tiempo» se multiplica por 50 para conseguir la temporización deseada, el resultado es asignado a la variable «pau- sa», utilizada como argumen- to.

En la línea 90 «PAUSE 0» de- tiene la ejecución del progra- ma hasta que se pulse la te- cla «S» o «N».

PROGRAMAS

El programa número «1» si- mula una máquina de escribir. los controles son.

- SYMBOL SHIFT + Q, borra el carácter anterior.
- SYMBOL SHIFT + W, provoca un retorno de carro.
- SYMBOL SHIFT + E, termina la edición de la página.

Una vez editada la página, puede sacarse una copia por impresora seleccionando la opción «L» o grabarse en cin- ta con «G». La opción «C» per- mite editar una nueva página o terminar.

La estructura es la siguien- te:

- 10 : Comentario con el nombre del pro- grama.
- 20 : Asignación del co- lor rojo para borde, azul para el fondo y blanco para los caracteres.

- 30 : Inicialización del cursor en la posi- ción 0, 0.
- 40 : Visualización del cursor. Se utiliza uno de los gráfi- cos predefinidos.
- 50 : Pausa hasta que se pulsa una tecla.
- 60-100 : Verificación de la tecla pulsada. Su código ASCII debe estar comprendi- do entre el núme- ro 32 (espacio) y el 122 (z) o ser uno de los códigos de control del progra- ma.

" < " (SYMBOL SHIFT + Q)
 " < > " (SYMBOL SHIFT + W)
 " > = " (SYMBOL SHIFT + E)

- 110 : Visualización de la tecla pulsada.
- 120-140 : Incremento de la posición del cur- sor y comproba- ción del final de lí- nea y final de pá- gina.
- 150 : Salto a la línea que visualiza el cursor
- 1000 : Empieza la rutina que borra el carác- ter situado a la iz- quierda del cursor (control = SYM- BOL SHIFT + Q).
- 1005 : Borra el cursor.
- 1010-1030 : Decremento de la posición del cur- sor y comproba- ción del principio de línea y princi- pio de página.
- 1040 : Salto a la línea que visualiza el cursor.
- 1100 : Comienzo de la ru- tina que hace sal- tar el cursor al co- mienzo de la si-

guiente línea, si- mulando el retor- no de carro de una máquina de escri- bir (control = SYMBOL SHIFT + W).

- 1105 : Borrado del cursor.
- 1110 : Inicialización de la nueva posición del cursor.
- 1120 : Salto a la línea que comprueba el final de página y línea.
- 1205 : Borrado del cur- sor.
- 1210 : Visualización de las opciones:
 L - Listar
 G - Grabar
 C - Continuar
- 1220-1250 : Comprobación de la opción elegida.
- 1260-1300 : Imprimir el conte- nido de la panta- lla. Se utiliza la sentencia «COPY»
- 1310 : Salto a la visuali- zación de las op- ciones.
- 1350-1380 : Grabar en cinta el contenido de la pantalla. Debe in- troducirse previa- mente el nombre que deseamos asignarle. Se uti- liza como argu- mento de «SAVE» la palabra clave «SCREEN\$».
- 1390 : Salto a la visuali- zación de opcio- nes.
- 1500-1550 : Decisión para edi- tar una nueva pá- gina o no.

El programa número «2» permite desplazar, con ayuda de unas teclas utilizadas co- mo cursor, un asterisco (*) a través de la pantalla, que va dejando un rastro de puntos (.) por donde va pasando.

PROGRAMA 1

```

10 REM *****
  * CURS0/BASIC *
  * *****
  * MAQUINA *
  * *****

20 BORDER 2: PAPER 1: INK 7: C
LS
30 LET X=0 LET Y=X
40 PRINT AT Y,X," "
50 PAUSE 0
60 LET A$=INKEY$
70 IF A$="<" THEN GO TO 1000
80 IF A$=">" THEN GO TO 1100
90 IF A$="=" THEN GO TO 1200
95 IF (A$=" " AND A$<="Z") TH
EN GO TO 110
100 GO TO 50
110 PRINT AT Y,X,A$
120 LET X=X+1
130 IF X=32 THEN LET X=0 LET Y
=Y+1
140 IF Y=22 THEN GO TO 1210
150 GO TO 40
1000 REM *****
  * BORRADO *
  * *****

1005 PRINT AT Y,X;" "
1010 LET X=X-1
1020 IF X=-1 THEN LET X=31 LET
Y=Y-1
1030 IF Y=-1 AND X=31 THEN LET Y
=0 LET X=0
1040 GO TO 40
1100 REM *****
  * RETORNO *
  * *****

1105 PRINT AT Y,X;" "
1110 LET X=0 LET Y=Y+1
1120 GO TO 140
1200 REM *****
  * ALMACENAR *
  * *****

```

Tenemos la posibilidad de elegir las teclas que vamos a utilizar como cursores. El programa tiene asignadas por defecto, es decir, si no se eligen otras, las siguientes:

- 7 - ARRIBA
- 6 - ABAJO
- 8 - DERECHA
- 9 - IZQUIERDA

La velocidad es otro de los parámetros que se puede elegir, su valor varía entre «1» (rápida) y «9» (lenta).

Aparte de las teclas utilizadas como cursor, existen otras dos que en combinación con SYMBOL SHIFT realizan una determinada función.

SYMBOL SHIFT + B, borra la pantalla.

SYMBOL SHIFT + C, fin del programa.

La estructura del programa es la siguiente:

10 : Comentario con el nombre del programa.

```

1205 PRINT AT Y,X," "
1210 PRINT #0;AT 1,1;"L-Lista /
G Graba / C-Continúa"
1220 IF INKEY$=" " THEN GO TO 120
0
1230 IF INKEY$="L" OR INKEY$="L"
THEN GO TO 1260
1240 IF INKEY$="G" OR INKEY$="G"
THEN GO TO 1350
1245 IF INKEY$="C" OR INKEY$="C"
THEN GO TO 1500
1250 GO TO 1220
1260 REM *****
  * IMPRESORA *
  * *****

```

```

1262 INPUT 0
1265 PRINT #0;AT 0,2;"Conecte la
impresora y pulse"
1270 PRINT #0;AT 1,10;"una tecla"

1280 PAUSE 0
1285 INPUT 0
1290 COPY
1300 PRINT #0;AT 1,6;"Impresion
terminada"; PAUSE 100
1310 GO TO 1210
1350 REM *****
  * GRABACION *
  * *****

```

```

1360 INPUT "Nombre (max 10 caracte-
ras) >>> ", LINE N$
1370 SAVE N$SCREEN$
1380 PRINT #0;AT 1,6;"Grabacion
terminada"; PAUSE 100
1390 GO TO 1210
1500 REM *****
  * CONTINUACION? *
  * *****

```

```

1510 INPUT 0
1520 PRINT #0;AT 1,0;"Quiere edi-
tar otra pagina (S/N)?"
1530 IF INKEY$="S" OR INKEY$="s"
THEN GO TO 10
1540 IF INKEY$="N" OR INKEY$="n"
THEN GOTO STOP
1550 GO TO 1530

```

20

30

40-90

100

Asignación del color rojo para el borde, verde para el fondo y negro para los caracteres.

: Llamada a la subrutina que presenta el menú con los cursores.

: Verificación de la opción elegida.

: Asignación de los valores por defecto.

PROGRAMA 2

```

10 REM *****
  *
  * CURSO/BASIC *
  *
  * *****
  *
  * MOVIMIENTO *
  *
  * *****
20 BORDER 2: PAPER 4. INK 0 C
L5
25 REM *****
  *
  * CAMBIO CURSOR *
  *
  * *****

30 GO SUB 1000
40 PRINT #0, AT 1,0, "Quiere ver
os cursores (S/N) >>>"
50 IF INKEY$="" THEN GO TO 50
60 LET k$=INKEY$
70 IF k$="N" OR k$="n" THEN GO
TO 100
80 IF k$="S" OR k$="s" THEN GO
SUB 1100 GO TO 110
90 GO TO 50
100 LET a$="7" LET b$="6" LET
d$="8" LET i$="5"
110 PAUSE 50
115 GO SUB 1300
120 CLS
125 REM *****
  *
  * RECUADRO *
  *
  * *****

130 FOR n=0 TO 31
140 PRINT AT 0,n, " "
150 NEXT n
160 FOR n=1 TO 20
170 PRINT AT n,0, " " AT n,31, " "
180 NEXT n
190 FOR n=0 TO 31
200 PRINT AT 21,n, " "
210 NEXT n
220 REM *****
  *
  * MOVIMIENTO *
  *
  * *****

230 LET posx=15
240 LET posy=11
250 GO SUB 1500
260 IF INKEY$="" THEN GO TO 260
270 IF INKEY$=a$ THEN PRINT AT
posy,posx, " " LET posy=posy+1
GO SUB 1500
280 IF INKEY$=b$ THEN PRINT AT
posy,posx, " " LET posy=posy+1
GO SUB 1500
290 IF INKEY$=d$ THEN PRINT AT
posy,posx, " " LET posx=posx+1
GO SUB 1500
300 IF INKEY$=i$ THEN PRINT AT
posy,posx, " " LET posx=posx-1
GO SUB 1500
310 IF INKEY$="*" THEN GO TO 34
0
320 IF INKEY$="?" THEN STOP
330 GO TO 260
340 GO SUB 1600
350 GO TO 230
1000 REM *****
  *
  * MOVIMIENTO *
  *
  * *****

```

```

1010 PRINT AT 2,0, " "
1020 PRINT AT 8,9, "7 - RÁPIDO,"
1030 PRINT AT 9,9, "8 - MEDIO,"
1040 PRINT AT 12,9, "9 - LENTA,"
1050 PRINT AT 15,9, "5 - IZQUIERDA,"
1060 PRINT AT 19,9, " "
1070 RETURN
1100 REM *****
  *
  * CURSORES *
  *
  * *****

```

```

1110 INPUT "Arriba >>> ", LINE a
$
1112 IF a$="" THEN GO TO 1110
1115 LET a$=a$(1)
1120 IF (a$<="0" AND a$<="9") OR
(a$<="a" AND a$<="z") OR (a$<="A"
AND a$<="Z") THEN PRINT AT 8,
9 a$ GO TO 1140
1130 GO TO 1110
1140 INPUT "Abajo >>> ", LINE b$
1142 IF b$="" THEN GO TO 1140
1145 LET b$=b$(1)
1150 IF ((b$<="0" AND b$<="9") O
R (b$<="a" AND b$<="z") OR (b$<="A"
AND b$<="Z")) AND b$<>a$ THE
N PRINT AT 9,9,b$: GO TO 1170
1155 GO TO 1140
1170 INPUT "Derecha >>> ", LINE
d$
1172 IF d$="" THEN GO TO 1170
1175 LET d$=d$(1)
1180 IF ((d$<="0" AND d$<="9") O
R (d$<="a" AND d$<="z") OR (d$<="A"
AND d$<="Z")) AND d$<>b$ AND
d$<>a$ THEN PRINT AT 12,9,d$ G
O TO 1200
1190 GO TO 1170
1200 INPUT "Izquierda >>> ", LIN
E i$
1202 IF i$="" THEN GO TO 1200
1205 LET i$=i$(1)
1210 IF ((i$<="0" AND i$<="9") O
R (i$<="a" AND i$<="z") OR (i$<="A"
AND i$<="Z")) AND i$<>b$ AND
i$<>d$ AND i$<>a$ THEN PRINT AT
15,9,i$ RETURN
1200 GO TO 1200
1300 REM *****
  *
  * VELOCIDAD *
  *
  * *****

```

```

1310 CLS
1320 PRINT AT 2,0, " "
1330 PRINT AT 8,9, "1 - RÁPIDO,"
1340 PRINT AT 9,9, " "
1350 PRINT AT 12,9, " "
1360 PRINT AT 15,9, "9 - LENTA,"
1370 PRINT AT 19,9, " "
1380 INPUT "Velocidad >>> ", vel
ocidad
1390 IF velocidad<1 OR velocidad
>9 THEN GO TO 1380
1410 RETURN
1500 REM *****
  *
  * VERIFICACION *
  *
  * *****

```

```

1510 IF posx<1 THEN LET posx=1
1520 IF posx>30 THEN LET posx=30
1530 IF posy<1 THEN LET posy=1
1540 IF posy>20 THEN LET posy=20
1550 PRINT AT posy,posx,"*"
1560 FOR n=1 TO velocidad
1570 NEXT n
1580 RETURN
1590 REM

```

```

*****
* BORRADO *
*
*****

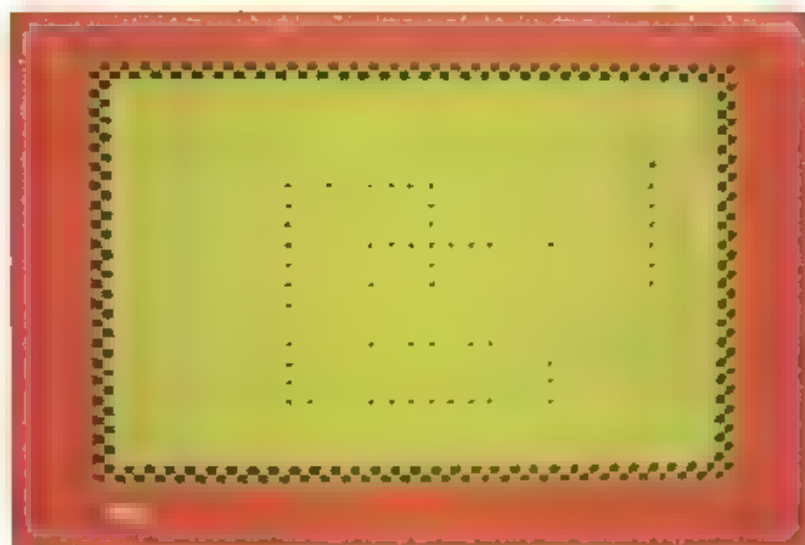
```

```

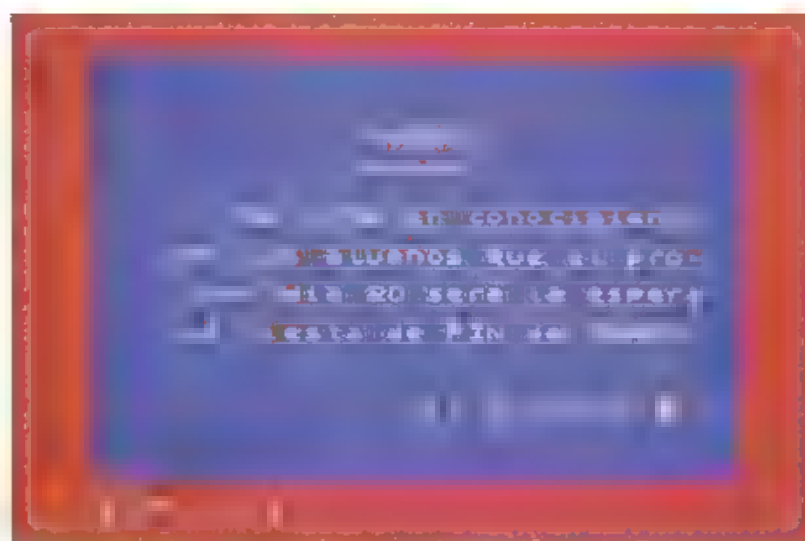
1610 FOR n=1 TO 20
1620 PRINT AT n,1,"*"
1630 NEXT n
1640 RETURN

```

- 110 : Retardo de un segundo aproximadamente. Llamada a la subrutina que pregunta la velocidad.
- 120 : Borrado de pantalla.
- 130-210 : Rutina que dibuja, con ayuda de los gráficos predefinidos, un recuadro.
- 230-240 : Inicialización de las coordenadas del asterisco.
- 250 : Llamada a la subrutina que dibuja el asterisco.
- 260-300 : Determinación de la tecla pulsada, cálculo de la nueva posición y llamada a la subrutina de visualización.
- 310 : Si la tecla pulsada es el asterisco (*), la opción de borrar es la elegida.
- 320 : Si la tecla pulsada es la interrogación (?), el programa termina su ejecución.
- 340 : Llamada a la subrutina de borrado.
- 1000-1070 Subrutina que visualiza el menú con la asignación de cursores inicial.
- 1100-1220 Subrutina utilizada para modificar la asignación inicial de cursores.
- 1300-1410 : Subrutina para la



Programa «Movimiento».



Programa «Máquina»

- Introducción de la velocidad.
- 1500-1580 : Verificación de que las coordenadas del asterisco se encuentran dentro del recuadro, visualización del mismo y temporización variable, dependiente de la velocidad elegida.
- 1600-1640 : Borrado de la parte interior del recuadro.

FUNCIONES

Existen una serie de funciones definidas dentro del BASIC, que pueden clasificarse en:

- Numéricas.
- De cadena.

Aparte de estas funciones, ya fijas, el usuario puede definir las suyas propias.

En general, una función proporciona un resultado después de haber efectuado unos cálculos con un dato denominado *parámetro*; éste forma parte del argumento de la instrucción. Para cada valor, la función retorna un resultado distinto.

Aunque no es necesario incluir los parámetros de una función entre paréntesis, es conveniente hacerlo, en algunos casos, por motivos de claridad. También hay que tener en cuenta que las funciones tienen mayor prioridad que las operaciones. Las funciones no se utilizan directamente como comandos, por el contrario necesitan ir acompañadas de las palabras clave «PRINT» (visualización), «LET» (asignación), «IF... THEN», etc.

Funciones numéricas

Estas pueden clasificarse en los siguientes grupos

- Matemáticas.
- Trigonómicas.
- Exponenciales.
- Logarítmicas.
- Aleatorias.
- Definidas.

ABS

Acceso al teclado

ABS



MODO E

}

Definición

Esta función retorna el *valor absoluto* del argumento.

Ejemplos.

- LET A = ABS 30
- PRINT ABS (-100 + 2)
- IF ABS valor < > 30 THEN...
- FOR n = 1 TO ABS X

El *valor absoluto* de una expresión se calcula despreciando su signo, por tanto, el resultado de las siguientes instrucciones será el mismo.

```
PRINT ABS 3542
PRINT ABS -3542
PRINT ABS +3542
```

En el siguiente programa se visualiza el *valor absoluto* de cualquier número comprendido entre 99999999 y -99999999.

```
10 REM *****
   VALOR ABSOLUTO
   ABS
   *****
20 INPUT Numero *** Numero
```

```
30 IF ABS Numero > 99999999 THEN
GO 20
40 IF Numero < 0 THEN LET AS = -54
50 GO TO 70
50 IF Numero < 0 THEN LET AS = PO
51 GO TO 70
60 LET AS = Negat 40
70 LET Absoluta = ABS Numero
80 FOR n = 5 TO 1 STEP 2
82 PRINT AT n 1
84 NEXT n
90 PRINT AT 5 0 Numero
100 PRINT AT 7 0 Signo
110 PRINT AT 9 0 Absoluta
120 GO TO 20
```

En la línea 30 se ha utilizado la función «ABS» para averiguar si el número estaba comprendido dentro del rango; si no, hubiéramos tenido que hacerlo de la forma:

```
30 IF Numero < -99999999 OR
Numero > 99999999 THEN
GOTO 20
```

INT

Acceso al teclado

INT



MODO E

VERIFY

Definición

La función «INT» retorna el *valor entero* de una expresión:

Ejemplos:

- LET X = INT Y
- PRINT INT (X + 3)
- IF INT n = n THEN ..
- FOR Z = INT t TO INT j

«NT» redondea *por defecto* el argumento, despreciando sus decimales. Redondear por defecto significa asumir el valor entero *inmediato inferior*, por tanto, se obtendrá el mismo resultado con cualquiera de las dos sentencias siguientes:

```
PRINT INT 3.00001
PRINT INT 3.99999
```

en ambos casos el resultado es «3».

Con los números negativos ocurre una cosa curiosa, ya que al efectuar el redondeo por defecto, aumenta su valor absoluto.

Ejemplo:

```
PRINT INT 3.00001
PRINT INT -3.99999
```

el resultado de ambas funciones es «-4».

El siguiente programa calcula si el número entero introducido por el teclado es *par* o *impar*.

```
10 REM *****
11 : PAR O IMPAR :
12 *****
20 INPUT "Numero >>> " numero
30 IF ABS numero > 99999999 OR INT numero < numero THEN GO TO 20
40 LET divisor=numero/2
50 LET entero=INT divisor
60 IF DIV 2 entero THEN LET
70 LET entero=entero
80 PRINT AT 5 0 "El numero ",
90 numero
100 GO TO 20
```

Con ayuda de la función «INT» se averigua en la línea 30 si la variable «número» tiene parte fraccionaria.

En el siguiente ejemplo se visualiza la parte entera y fraccionaria del número positivo que se introduce por teclado.

las pequeñas diferencias que pueden existir, entre las partes fraccionarias son debidas a los cálculos.

```
10 REM *****
11 : PARTE ENTERA :
12 : INT :
13 *****
20 INPUT "Numero >>> " numero
30 IF numero < 0 OR numero > 99999
40 THEN GO TO 20
50 LET entero=INT numero
60 LET fraccionario=numero-ent
70
80 FOR n=5 TO 9 STEP 2
90 PRINT AT n 10
100 NEXT n
110 PRINT AT 5 0 "Numero "
120 PRINT AT 7 0 "Parte entera
130 PRINT AT 9 0 "Parte decimal
140 GO TO 20
```

Ejemplos:

- PRINT SGN -30
- LET a = SGN C
- PRINT SGN (7 * (-5))
- LET X = 7 * SGN Z

El siguiente programa nos indica si el número introducido por teclado es positivo o negativo.

```
10 REM *****
11 : SIGNO :
12 : "SGN" :
13 *****
20 LET positivo=1
30 LET negativo=-1
40 LET cero=0
50 INPUT "Numero >>> " numero
60 IF ABS numero > 99999999 THEN
70 GO TO 50
80 IF SGN numero=positivo THEN
90 LET a=positivo GO TO 100
100 IF SGN numero=negativo THEN
110 LET a=negativo GO TO 100
120 LET a=cero
130 PRINT AT 5 0 "El numero ",
140 numero
150 GO TO 20
```

SQR

Acceso al teclado

SQR

H GOSUB MODO F
CIRCLE

Definición

La función «SQR» calcula la raíz cuadrada del argumento.

Ejemplos:

- PRINT SQR 144
- LET r = SQR 625 + 13
- PRINT SQR raíz
- LET n = SQR (25 + a)

«SQR» sólo calcula raíces de tipo *real*, si se pretende calcular una *imaginaria*

Acceso al teclado

SGN

F FOR MODO E

Definición

La función «SGN» nos indica qué signo tiene la expresión que estamos evaluando. Los posibles valores que puede retornar son:

- 1 Si es positiva.
- 1 Si es negativa.
- 0 Si es 0

($\sqrt{-144}$) se visualizará el mensaje de error.

A Invalid argument

Ejemplo:

— PRINT SQR 144

Cuando se evalúa una variable, ésta puede tomar un valor negativo, para asegurarnos que al calcular su raíz no nos dé error, podemos utilizar la función «ABS».

```
10 LET raíz = -144
20 PRINT SQR (ABS raíz)
```

Aunque el Spectrum sólo tiene una función de radicación (SQR), cuyo índice es 2 se pueden obtener raíces de cualquier orden, para ello hay que basarse en la siguiente igualdad.

$$\sqrt[n]{r} = r^{1/n}$$

donde «r» es el radicando e «n» el índice.

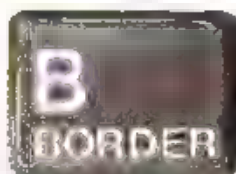
Ejemplo:

— Raíz cúbica de 27
PRINT 27 \uparrow (1/3)
— Raíz quinta de 32
PRINT 32 \uparrow (1/5)

BIN

Acceso al teclado

BIN



MODO E

BRIGHT

Definición

Permite la representación de los números en notación binaria, para una mayor aclaración conviene consultar el capítulo «CONSTANTES Y VARIABLES» (Pág. 30).

Ejemplos:

```
— LET a = BIN 1001
— PRINT —BIN 11011.2
LET c = BIN 1101na
— PRINT SQR BIN 1111
```

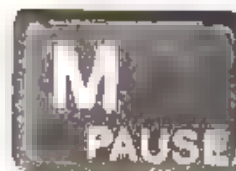
El mayor número que se puede representar en este tipo de notación, es «65535».

En la página 25 se presenta el programa «CODEBIN» que realiza la transformación inversa, es decir, pasar de notación decimal a binaria.

PI

Acceso al teclado

PI



MODO E

INVERSE

Definición

«PI» es el nombre de una letra griega (π) usada como constante en multitud de cálculos matemáticos. Su valor aproximado es.

$$\pi \approx 3,14159265...$$

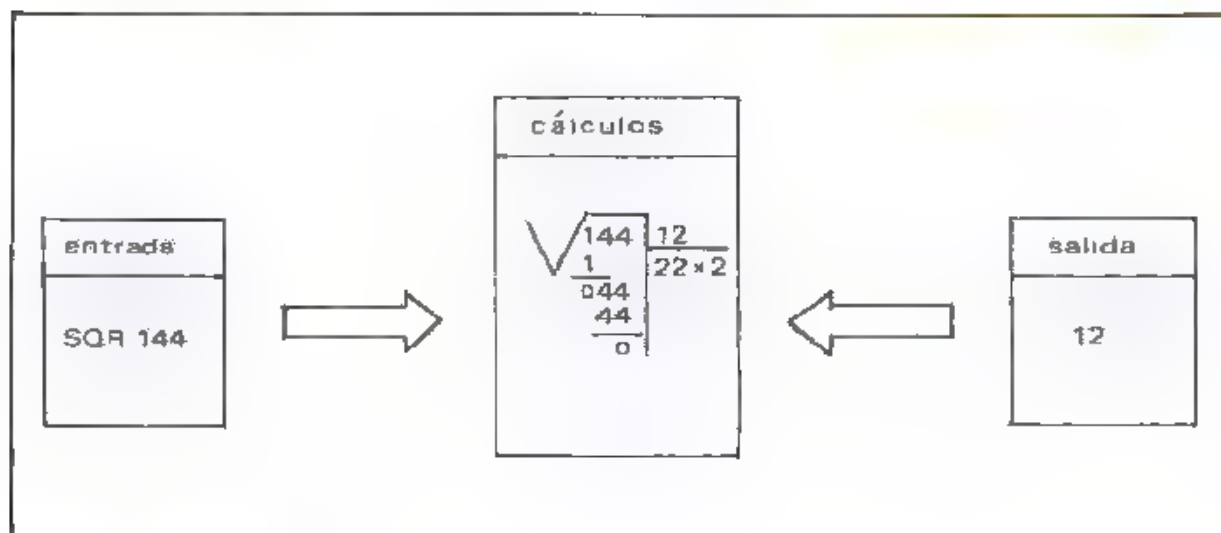
Ejemplos:

a) Cálculo de la longitud de una circunferencia de radio 6.

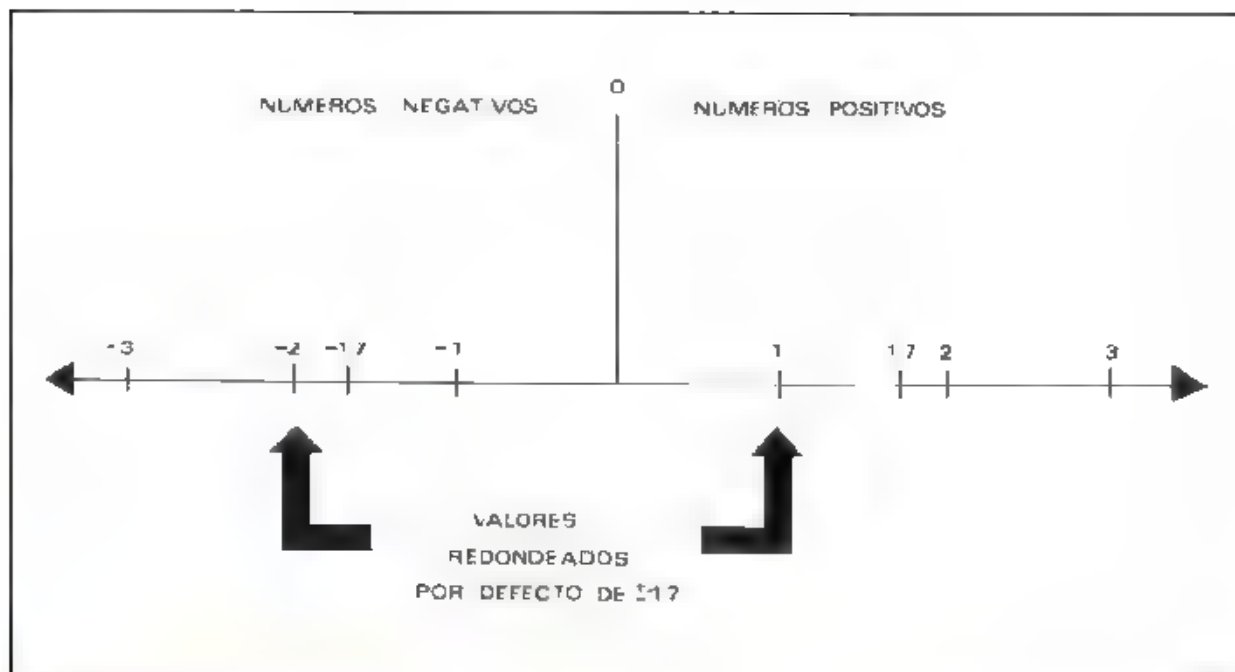
```
10 LET radio = 6
20 PRINT 2 * PI * radio
```

b) Cálculo de la impedancia que presenta una bocina de 0.5 Henrios a la frecuencia de 12 KHz.

```
10 LET L = 0,5
20 LET F = 12000
```



Ejemplo «Función».



Función «INT».

```
30 LET Z = 2 * PI * F * L
40 PRINT Z
```

El radián

Es una unidad de medida de ángulos. Se puede definir como el ángulo cuyo arco tiene la misma longitud que el radio de la circunferencia (ver figura)

La equivalencia entre el radián y los grados sexagesimales es

$90^\circ = \pi/2$ radianes
 $180^\circ = \pi$ radianes
 $270^\circ = 3\pi/2$ radianes
 $360^\circ = 2\pi$ radianes

por tanto:

$$1 \text{ radian} = \frac{180}{\pi}$$

Para calcular en el Spectrum el seno, coseno o tangente de un ángulo, este deberá ser expresado en radianes. Para transformar grados a radia-

nes deberá usarse la siguiente fórmula.

$$\text{radianes} = \frac{\text{grados} \cdot \pi}{180}$$

y para transformar a la inversa:

$$\text{grados} = \frac{\text{radianes} \cdot 180}{\pi}$$

El siguiente programa transforma grados, minutos y segundos en radianes:

```
10 REM *****
   :      RADIANS
   : *****
20 INPUT "Grados : " : grados
30 IF grados<0 OR grados>360 THEN GO TO 20
35 CLS
40 PRINT "Grados = .. ,grados
50 INPUT "Minutos : " : minutos
60 IF minutos<0 OR minutos>60 THEN GO TO 20
70 PRINT "Minutos = .. ,minutos
80 INPUT "Segundos : " : segundos
90 IF segundos<0 OR segundos>60 THEN GO TO 20
100 PRINT "Segundos = .. ,segundos
110 LET total=grados+minutos/60+segundos/3600
120 IF total>360 THEN CLS GO TO 20
130 LET radianes=total*PI/180
140 PRINT "Radianes = .. ,radianes
150 GO TO 20
```

Funciones trigonométricas

SIN

Acceso al teclado

SIN



MODO

ASN

Definición

«SIN» calcula el seno de un ángulo expresado en radianes.

Ejemplos:

- PRINT SIN (90 * PI/180)
- LET C = SIN 1
- PRINT SIN (270 * PI/180)
- LET C = SIN ángulo

En un triángulo rectángulo, el seno de un ángulo es la razón que existe entre el cateto opuesto y la hipotenusa.

Para un ángulo comprendido entre «0» y «180» grados el valor del seno es positivo, y negativo entre «180» y «360».

COS

Acceso al teclado

COS



MODO E

ACS

Definición

La función «COS» calcula el coseno de un ángulo, éste debe estar expresado en radianes.

Ejemplo:

- LET n = COS 2
- PRINT COS (17 * PI/180)
- LET valor = COS total
- PRINT COS (320 * PI/180)

El coseno de uno de los ángulos de un triángulo rectángulo es la razón que hay entre el cateto adyacente y la hipotenusa.

El valor del seno es positivo para un ángulo comprendido entre «0» y «90» o entre «270» y «360». Es negativo entre «90» y «270».

TAN

Acceso al teclado

TAN



MODO E

ATN

Definición

«TAN» retorna la *tangente* de un ángulo expresado en radianes.

Ejemplos.

- PRINT TAN (45 * PI/180)
- LET C = TAN a fa
- PRINT TAN 1
- LET d = TAN (beta + 2)

La tangente de un ángulo es la razón que hay entre el cateto opuesto y el cateto adyacente, de un triángulo rectángulo.

ASN

Acceso al teclado

SIN



MODO E

ASN

Definición

La función «ASN» calcula el arcoseno, es decir, el valor de un ángulo a partir de su seno. El valor retornado está expresado en radianes.

Ejemplos.

- LET a = ASN 0,5
- PRINT 180/PI * ASN 1
- LET C = ASN (alfa) * 180/PI
- PRINT ASN 0,7

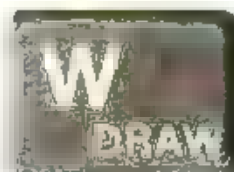
El valor del argumento debe estar comprendido entre +1 y -1, de lo contrario, se visualizará el mensaje de error.

A invalid argument

ACS

Acceso al teclado

COS



MODO E

ACS

Definición

«ACS» es la función que retorna el *arcoseno*, ángulo calculado a partir del coseno. Las unidades son expresadas en radianes.

Ejemplos:

- LET d = ACS 0,9
- PRINT ACS (0,35) * 180/PI
- LET K = ACS beta
- PRINT ACS alfa

Al igual que la función «ASN», el argumento de «ACS» debe estar comprendido dentro del rango de +1 y -1.

ATN

Acceso al teclado

TAN



MODO E

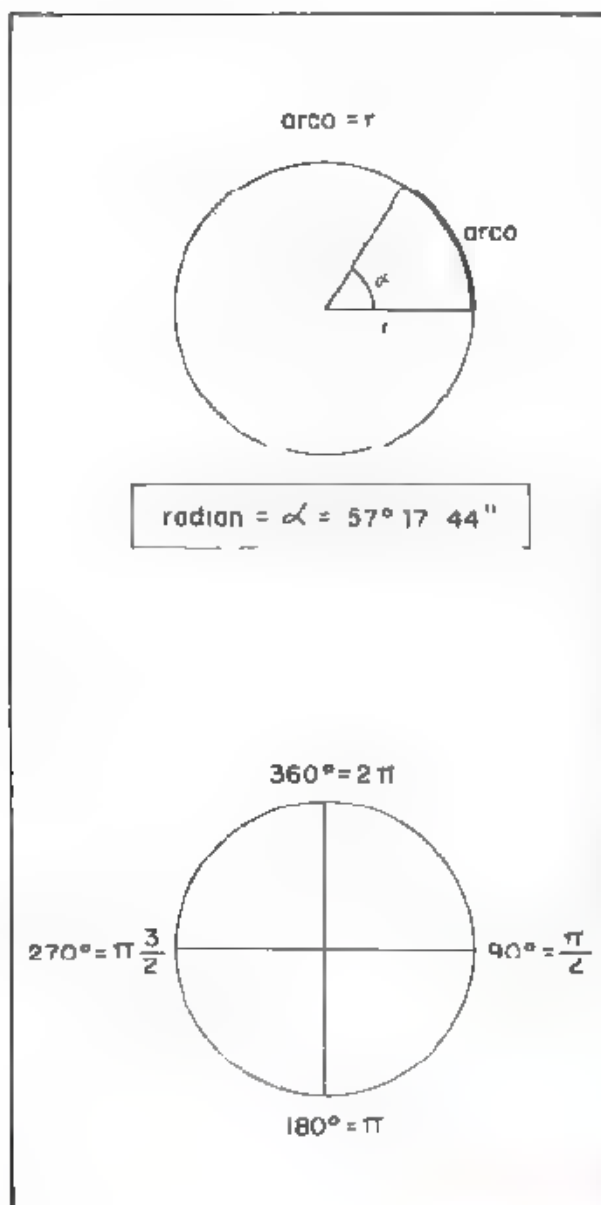
ATN

Definición

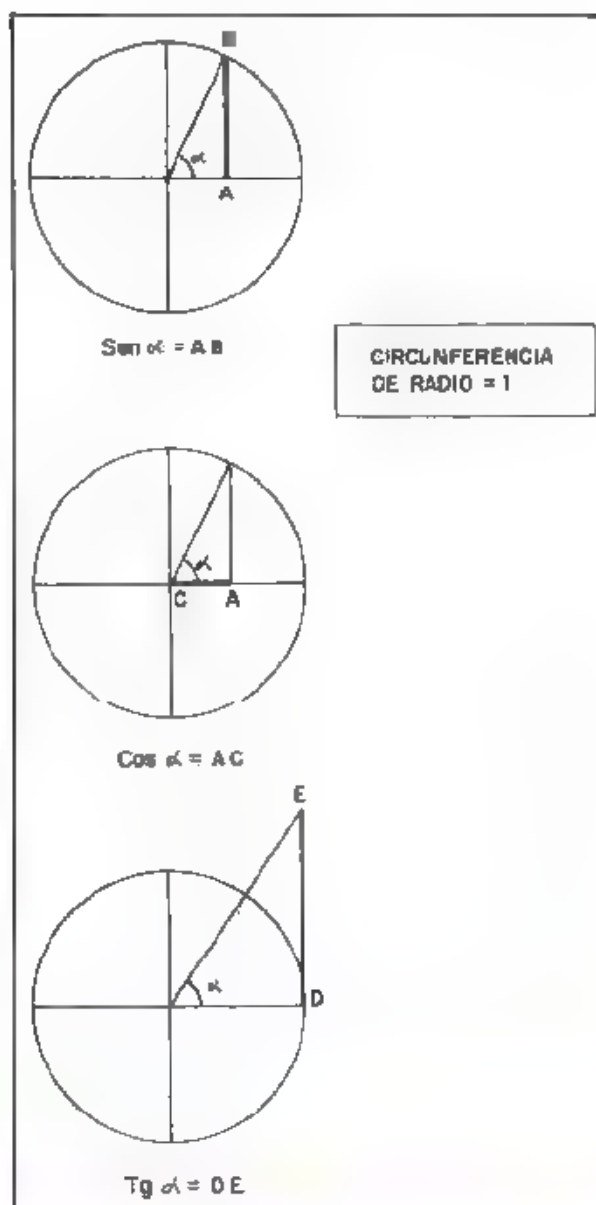
La función «ATN» calcula el *arctangente*, es decir, el valor de un ángulo en radianes, a partir de su tangente.

Ejemplo:

- LET r = ATN gamma
- PRINT ATN 10
- LET C = 180/PI * ATN n
- PRINT ATN (10 + alfa)



Radianes



Funciones trigonométricas

Aplicación de la trigonometría

Los programas que a continuación se presentan, son una pequeña muestra de las múltiples aplicaciones de la *trigonometría*

- a) Calcular la altura que alcanza una escalera apoyada en una pared, conociendo su longitud y el ángulo que forma con el suelo.

```
10 REM *****
11 REM      ESCALERA
12 REM *****
```

```
20 BORDER & PAPER & INK 0 C
30 INPUT Longitud escala D
32 CLS
40 IF D <= 0 THEN GOTO 10
42 PRINT Longitud D
50 INPUT Inc inclinacion grados
52 IF grados <= 0 THEN GOTO 10
54 PRINT Inc inclinacion grados
60 GO SUB 1000
62 LET A = D * SIN(grados*PI/180)
64 PRINT Altura A
70 GO TO 30
1000 REM *****
1010 LET A = D * SIN(PI/180)
1020 RETURN
```

- b) Calcular la altura de una torre, conociendo la distancia que nos separa de ella y la visual hasta su parte más alta.

```
10 REM *****
11 REM      TORRE
12 REM *****
20 BORDER & PAPER & INK 0 C
30 INPUT Dis. entre obs. y torre
32 CLS
40 IF D <= 0 THEN GOTO 10
42 PRINT Dis. entre obs. y torre
50 INPUT visual grados
52 IF grados <= 0 THEN GOTO 10
54 PRINT visual grados
60 GO SUB 1000
62 LET A = D * TAN(visual*PI/180)
64 PRINT altura A
70 GO TO 30
1000 REM *****
1010 LET A = D * TAN(PI/180)
1020 RETURN
```

- c) Calcular la longitud de la sombra que proyectará un árbol, conociendo su altura.

ra y el ángulo que forma el sol con el horizonte

```

10 REM *****
11 REM      APOLO
12 REM *****
20 BORDER 4 PAPER 4 TYP 3 C
30 INPUT A I 3 ARBO RA 5
40 IF A I 3 ARBO RA 5 THEN
50 PRINT A I 3 ARBO RA 5
60 INPUT I 3 ARBO RA 5
70 IF I 3 ARBO RA 5 THEN
80 PRINT I 3 ARBO RA 5
90 GOTO 1000
100 LET I 3 ARBO RA 5 = 1000
110 PRINT I 3 ARBO RA 5
120 GOTO 1000
130 REM *****
140 REM      APOLO
150 REM *****
160 GOTO 1000
170 REM *****
180 REM      APOLO
190 REM *****
200 GOTO 1000

```

Función exponencial

Se llaman así aquellas funciones en las que el exponente es un número variable.

Ejemplo:

$$y = a^x$$

donde «a» es la base y «x» el exponente variable.

Cuando la base es mayor que uno ($a > 1$), la función exponencial es una *función creciente*

Ejemplo:

```

10 LET base = 3
20 FOR X = 1 TO 22
30 PRINT base ^ X
40 NEXT X

```

Observe cómo los diversos valores que toma la función de base «3» y exponente variable, entre «1» y «22», van aumentando

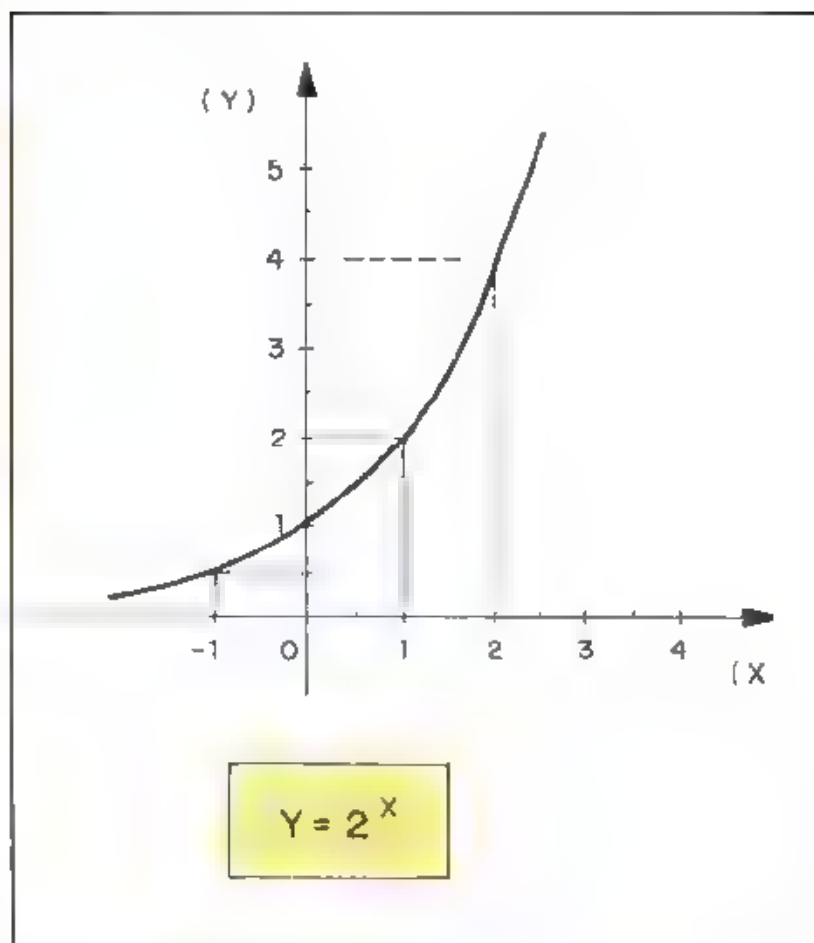
Cuando por el contrario, el valor de la base se halla comprendido entre 0 y 1 ($0 < a < 1$), se trata de una *función decreciente*.

Ejemplo:

```

10 LET base = 0.99
20 FOR X = 1 TO 22
30 PRINT base ^ X
40 NEXT X

```



$$Y = 2^X$$

Función exponencial creciente

En este caso se observa que el valor de la función va decreciendo.

De los diversos valores que puede tomar la base «a», hay uno que se utiliza frecuentemente en Matemáticas, Física y otras ciencias, es el denominado *número «e»*, base de los *logaritmos neperianos* y cuyo valor aproximado es:

$$e = 2.7182818$$

EXP

Acceso al teclado

EXP



MODOS E

Definición

«EXP» retorna el valor de la función exponencial creciente, teniendo como base el número «e»

Ejemplo:

```

— LET c = EXP 4
— PRINT EXP total
— LET k = c/EXP 8
— PRINT EXP (k + 3)

```

La función «EXP» cumple la siguiente igualdad:

$$\text{EXP } x = e^x = e \wedge x$$

Una forma de visualizar el valor numérico de «e», es:

```
PRINT EXP 1
```

ya que todo número elevado al

exponente unidad da como resultado el propio número:

$$a^1 = a$$

Ejemplo.

PRINT 27 ↑ 1

Función logarítmica

Es la inversa de la función exponencial

Ejemplo:

$$y = \log_a x$$

La expresión anterior se lee de la siguiente forma: «y» es igual al logaritmo de «x» en base «a»; donde «a» es la base de los logaritmos.

Se denomina logaritmo de un número a, exponente a que es preciso elevar la base para obtener dicho número.

$$\log_a x = y$$

$$a^y = x$$

Las bases de los logaritmos más utilizadas son la *decimal* y la *neperiana*. Los logaritmos de base decimal son también conocidos como logaritmos *vulgares* o de *Briggs*.

Ejemplo

$$y = \log_{10} x$$

Los logaritmos neperianos deben su nombre a John Neper (matemático inglés) y tienen como base e conocido número «e». También son llamados logaritmos *naturales*.

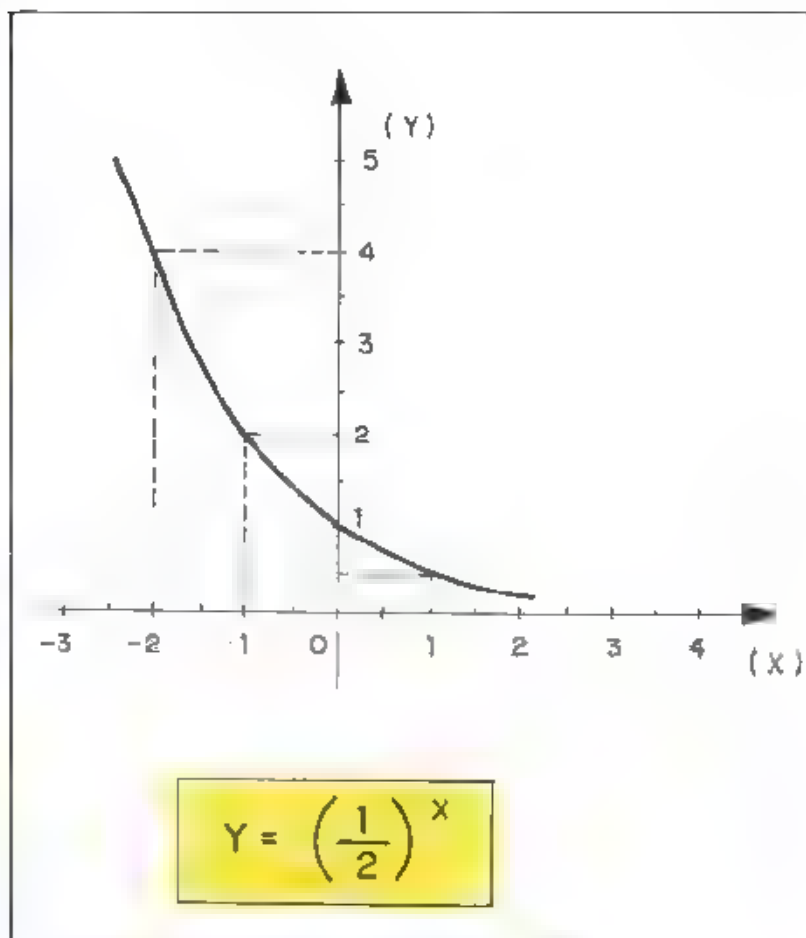
Ejemplo:

$$y = \log_e x$$

abreviadamente, puede escribirse el logaritmo neperiano como «ln».

Ejemplo:

$$y = \ln x$$



Función exponencial decreciente

La función logarítmica incluida en el juego de sentencias del Spectrum, es la de base neperiana.

Ejemplos:

- LET a = LN 7
- PRINT LN suma
- LET b = 2 * LN k
- PRINT LN (17/n)

Cuando el argumento es igual o inferior a «0», se visualiza el mensaje:

A Invalid argument

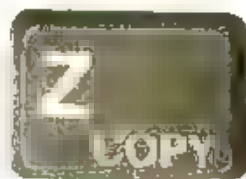
La utilización de los logaritmos es de uso frecuente en calculadoras de las denominadas científicas, y como no, en el Spectrum, ya que basándose en dos de los teoremas de dicha función se realizan los cálculos internos de potenciación y radicación.

- a) El logaritmo de una potencia es igual al producto del exponente por el logaritmo de la base.

LN

Acceso al teclado

LN

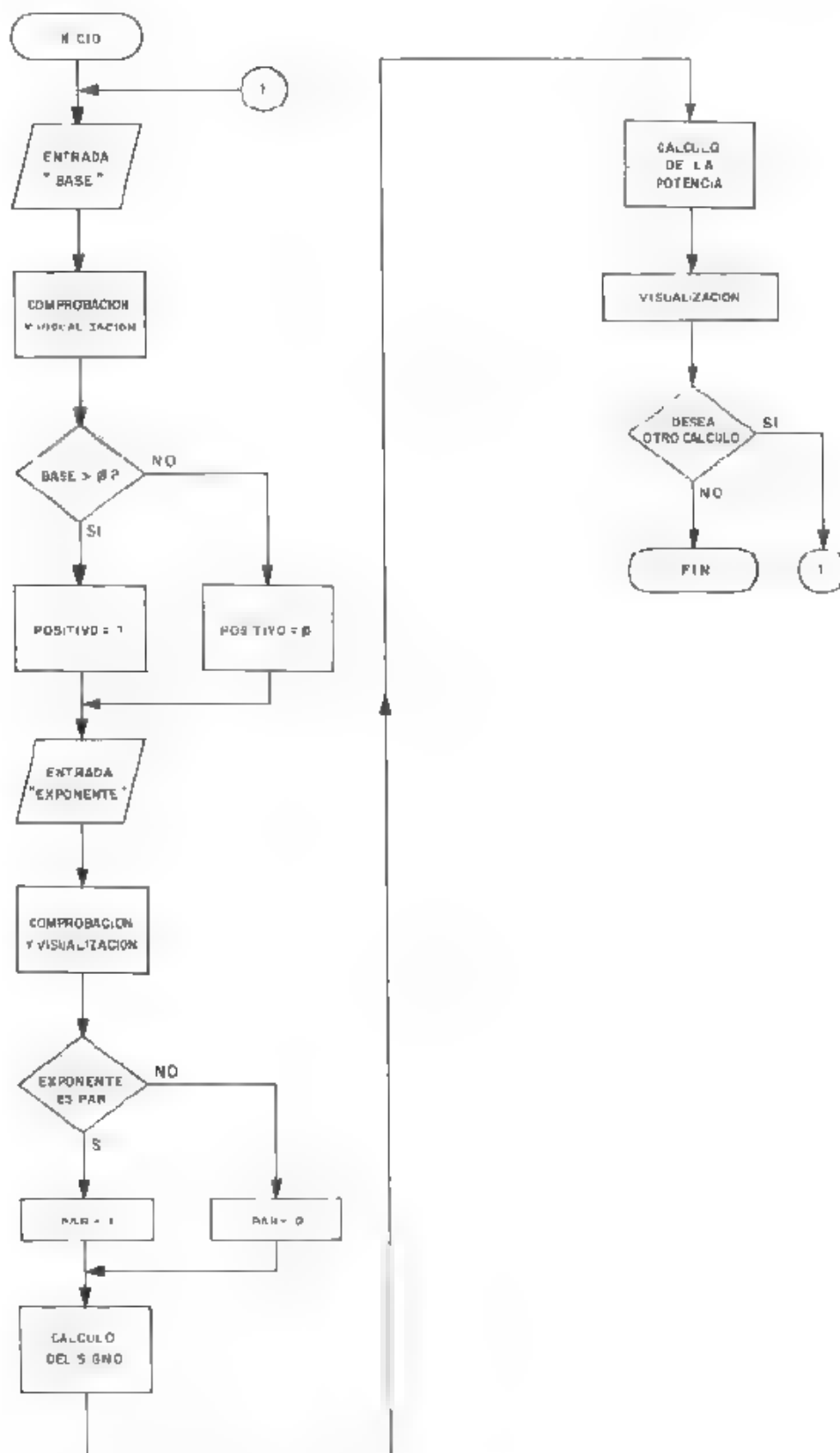


MODO E

BEEP

Definición

La función «LN» retorna el logaritmo neperiano del argumento.



Programa «potencia».

PROGRAMA 1

```

10 REM *****
   *
   * POTENCIACIÓN *
   *
   *****

20 INPUT "Base >>> "; base
30 IF (ABS base>999) OR (base=
0) THEN GO TO 20
40 PRINT "Base .....", bas
e
50 IF base>0 THEN LET positivo
-1: GO TO 70
60 LET positivo=0
70 INPUT "Exponente >>> "; exp
onente
80 IF ABS exponente>10 OR INT
exponente<>exponente THEN GO TO
70
90 PRINT "Exponente .....", exp
onente
100 LET divisor=exponente/2
110 LET entero=INT divisor
120 IF divisor=entero THEN LET
par=1: GO TO 140
130 LET par=0
140 IF (positivo=0 AND par=1) O
R (positivo=1) THEN LET s$="+":
GO TO 160
150 LET s$="-"
160 LET resultado=EXP (exponent
e*LN (ABS base))
170 PRINT base;" elevado a "; ex
ponente;" es igual a:"
180 PRINT " "s$;" " resultado
190 PRINT #0;"Desea otro calcul
o (S/N)?"
200 IF INKEY$="" THEN GO TO 200
210 LET a$=INKEY$
220 IF a$="s" OR a$="S" THEN CL
S: GO TO 10
230 IF a$="n" OR a$="N" THEN CL
S: STOP
240 GO TO 210

```

$$\ln a^b = b \cdot \ln a$$

por tanto

$$a^b = \text{exponencial}(b \cdot \ln a)$$

Para comprobar que el ordenador realiza la potenciación con ayuda de los logaritmos, introduzca el siguiente comando directo:

$$\text{PRINT } 2 \uparrow 13$$

observará que el resultado es «8192»; si restamos ambos valores, el resultado debería ser «0».

$$\text{PRINT } 2 \uparrow 13 - 8192$$

¿Por qué razón no es «0»? esto se debe a que el ordenador

no multiplica «13» veces el número «2», sino que por el contrario, lo calcula de acuerdo con el teorema expuesto anteriormente.

Introduzca:

$$\text{PRINT EXP (13 * LN 2)}$$

realice la siguiente operación

$$\text{PRINT } 2 \uparrow 13 - (\text{EXP (13 * LN 2)})$$

en este caso el resultado sí es cero.

Al realizar la operación $2 \uparrow 13$ no se visualiza «8192.000022888184» ya que, debido al sistema de presentación del Spectrum, este valor queda redondeado a «8192».

- b) El otro teorema dice que el logaritmo de una raíz es igual al cociente entre el logaritmo del radicando y el índice de la raíz.

$$\ln \sqrt[b]{a} = \ln(a)/b$$

luego

$$\sqrt[b]{a} = \text{exponencia}(\ln(a)/b)$$

Al igual que en el caso anterior, la radicación también se realiza con ayuda de los logaritmos.

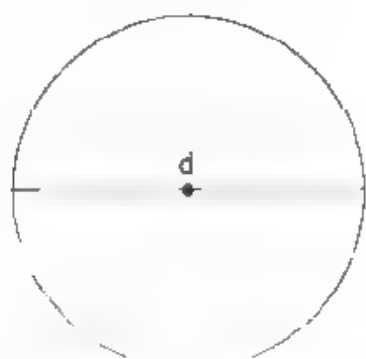
Realice los siguientes pasos:

```

PRINT SQR 144
PRINT SQR 144 - 12
PRINT EXP (LN 144/2)
PRINT SQR 144 - (EXP (LN 144/2))

```

Las explicaciones dadas en el caso de la potenciación también son válidas para la potenciación

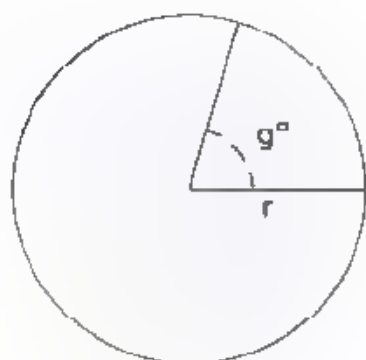


AREA DEL CIRCULO

$$S = \pi \left(\frac{d}{2} \right)^2$$



```
DEF FN s(d)=PI*(d/2)^2
```



AREA DEL SECTOR
CIRCULAR

$$S = \frac{\pi r^2}{360} \cdot g^\circ$$



```
DEF FN c(r,g)=PI*r^2/360*g
```

Definición de funciones

Compare los resultados anteriores con los proporcionados por «144 ÷ (1/2)».

Aunque el Spectrum solamente calcula, de forma directa, los logaritmos de base neperiana, pueden calcularse en cualquier otra base, teniendo en cuenta la siguiente igualdad

$$\log_a X = \frac{\ln X}{\ln a}$$

que dice, que el logaritmo de un número «X» en base «a» es igual a, cociente entre el logaritmo neperiano del número y el logaritmo neperiano de la base.

Ejemplos:

- a) Calcular el logaritmo en base 10 de 4 ($\log_{10} 4$).

```
PRINT LN(4)/LN(10)
```

- b) Calcular el logaritmo de 15 en base 2 ($\log_2 15$).

```
PRINT LN 15/LN 2
```

El siguiente programa calcula el logaritmo de cualquier número comprendido entre «1» y «99999999», la base puede elegirse entre «2» y «255».

```

10 REM *****
   LOGARITMO
*****
20 INPUT Base; "Base: "; Base
30 IF (Base = 1 OR Base > 255) THEN
   GO TO 20
40 INPUT Numero; "Numero: "; Numero
50 IF (Numero < 1 OR Numero > 99999999) THEN
   GO TO 50
60 PRINT "Logaritmo de "; Numero
70 LET Loga = LOG(Numero) / LOG(Base)
80 PRINT "Logaritmo de "; Numero
90 PRINT "en base "; Base " es: "; Loga
100 GOTO 10

```

Debido a que las potencias se calculan con ayuda de los logaritmos neperianos, no se puede calcular una potencia de base negativa, ya que daría el mensaje de error:

A Invalid argument

Ejemplo:

```
PRINT ( 2 ) 3
```

Con ayuda del programa número «1» se puede calcular hasta la potencia décima de cualquier número comprendido entre «999» y «-999», excepto el cero.

Definición de funciones

El sistema operativo del Spectrum permite al usuario definir sus propias funciones, éstas pueden definirse en cualquier parte del programa, pero por motivos de claridad conviene hacerlo al comienzo del mismo.

Para poder utilizar estas funciones posteriormente, es necesario que tengan asignado

un nombre, éste está compuesto por una sola letra en el caso de funciones numéricas y por una letra seguida del signo «\$» en el caso de funciones de cadena.

DEF FN

Acceso al teclado

EDIT



MODO E

DEF FN

Definición

Esta sentencia permite la definición de las funciones de usuario.

Su estructura general es:

SENTENCIA	ARGUMENTO
DEF FN	nombre (variable) = función

Ejemplos:

```

DEF FN a (X) = 7 * X + SIN X
DEF FN c (y) = COS (y - 200)
DEF FN j (n) = PI + n
DEF FN t (u) = 20 * (2 + J)

```

Las variables también tienen que estar formadas por una sola letra.

En los ejemplos anteriores se incluía una sola variable en la definición de la función, pero ésta puede contener hasta veintiséis distintas, una por cada letra.

Ejemplos:

```

DEF FN n (a, b) = (a + b) / 100
DEF FN r (l, m, n) = EXP n * (mm + LN l)

```

Llamadas a funciones definidas



Acceso al teclado

RED
CAPS LOCK



MODO E

FN

Definición

Esta sentencia permite utilizar las funciones definidas previamente.

Su estructura general es:

FN nombre (parámetros)

Ejemplos:

```

PRINT FN a (10)
LET x = FN n (total)
PRINT FN X (37,23)
LET j = FN p (c, 3)

```

Veamos unos ejemplos prácticos:

a) Calcular el área de un círculo en función de su diámetro. Primero es necesario definir la función

```
10 DEF FN S (d) = PI * (d/2) ^ 2
```

Veamos los valores que retorna la función para distintos diámetros (entre 1 y 20).

```

20 FOR n = 1 TO 20
30 PRINT "Diámetro: "; n
   PRINT "Área: "; FN S (n)
40 NEXT n

```

b) Calcular el área de un sector circular en función de su radio y del número de grados.

Definición de la función:

```
10 DEF FN C (r,g) = PI * r ^ 2/360  
  * g
```

Un ejemplo de utilización podría ser el siguiente:

```
20 INPUT "Rad o. ";radio  
30 INPUT "grados"  grados  
40 PR NT FN C radio,grados
```

Ejemplo.

```
10 REM *****  
      ERROR P  
      *****  
20 INPUT "temperatura en "  error  
30 LET "FN C" = valor
```

- b) Cuando se hace una llamada a una función definida, y no coincide el número de parámetros aparece el mensaje:

Q Parameter error

Este mensaje también aparece cuando alguno de los parámetros no es del mismo tipo que los definidos en la función; es decir, un parámetro de cadena en

una función numérica o viceversa.

Ejemplo:

```
10 REM *****  
      ERROP Q  
      *****  
20 DEF FN C C V = " " C V  
30 INPUT "Numero 1"  n1  
40 INPUT "Numero 2"  n2  
50 PRINT FN C n1
```

en este caso falta un parámetro en la llamada de la función. También daría error las siguientes instrucciones:

```
50 PRINT FN r ,n1, n2, 10)
```

(sobra un parámetro)

```
50 PRINT FN r ,n1 "pepe")
```

(parámetro de cadena)

Errores

Hay una serie de mensajes típicos de error que se visualizan al manejar incorrectamente las funciones.

- a) Al intentar manejar una función no definida previamente aparece el mensaje:

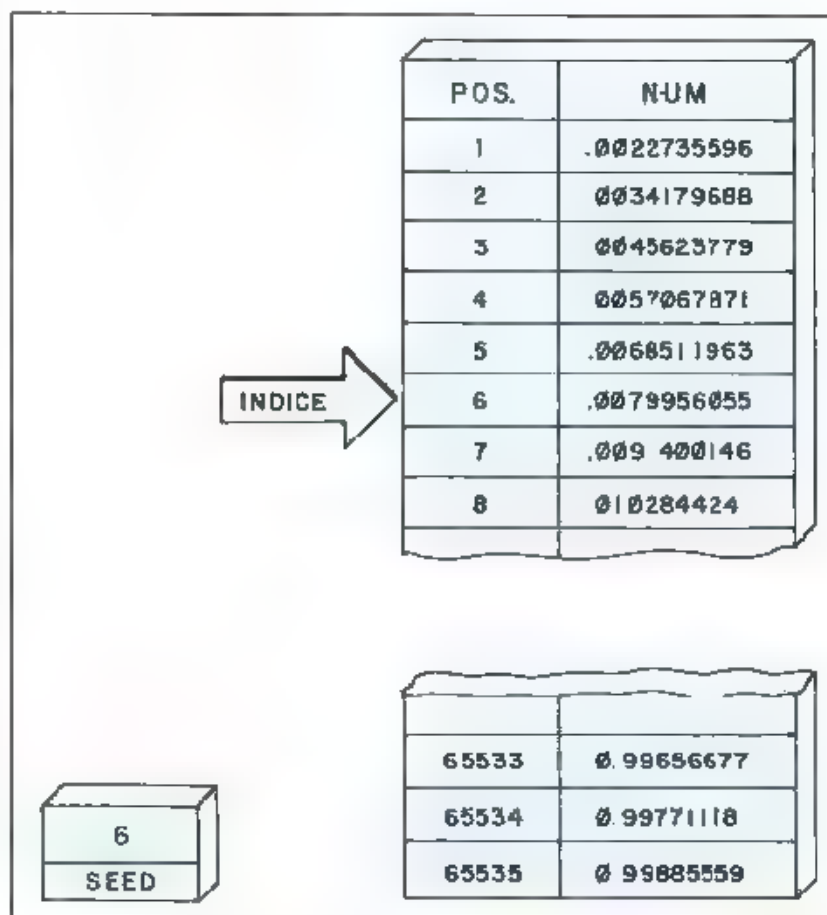
P FN without DEF

FUNCION ALEATORIA

Como su propio nombre indica, una función aleatoria es aquella que retorna un valor al azar, es decir, aquel que no sigue ninguna ley o algoritmo; en realidad esto no es del todo cierto ya que es bastante difícil, por no decir imposible, implementar una función de este tipo en un ordenador, por tanto, éstos incluyen una función llamada *pseudoaleatoria*.

La función pseudoaleatoria, siguiendo un algoritmo, elige adecuadamente ciertos números para que parezcan aleatorios. En el caso del Spectrum existe una secuencia cíclica formada por «65536» números distintos.

Las funciones aleatorias tienen un extenso campo de aplicación en los juegos y en los programas didácticos.



RND

Variable «SEED».

Acceso al teclado

RND



MERGE

MODO **E**

Definición

«RND» retorna uno de los

«65536» números que forman la secuencia de números aleatorios. Al ser una función, debe ir acompañada de sentencias tales como «PRINT», «LET», «IF», etc.

Ejemplos

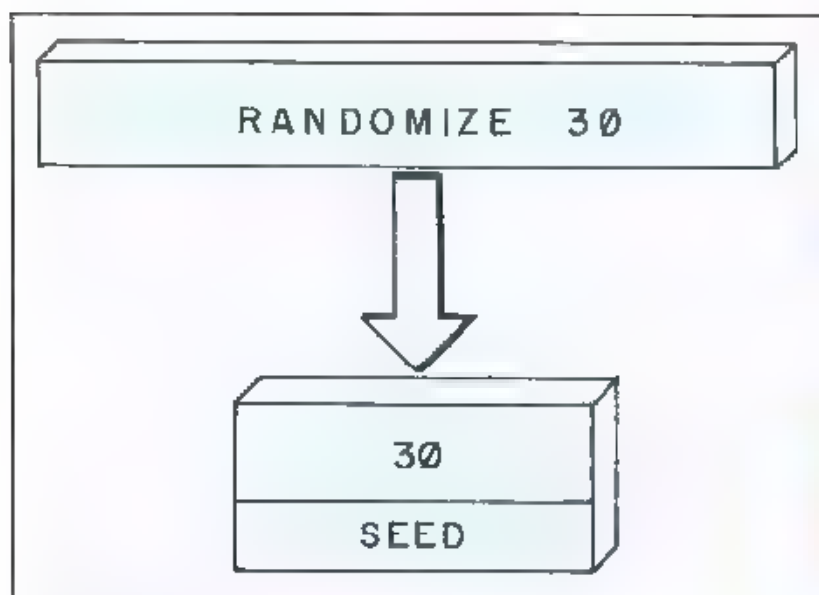
```
LET a = RND
- LET b = INT (RND + 5)
PRINT RND
PRINT 3 + RND
```

Ejecute el siguiente programa:

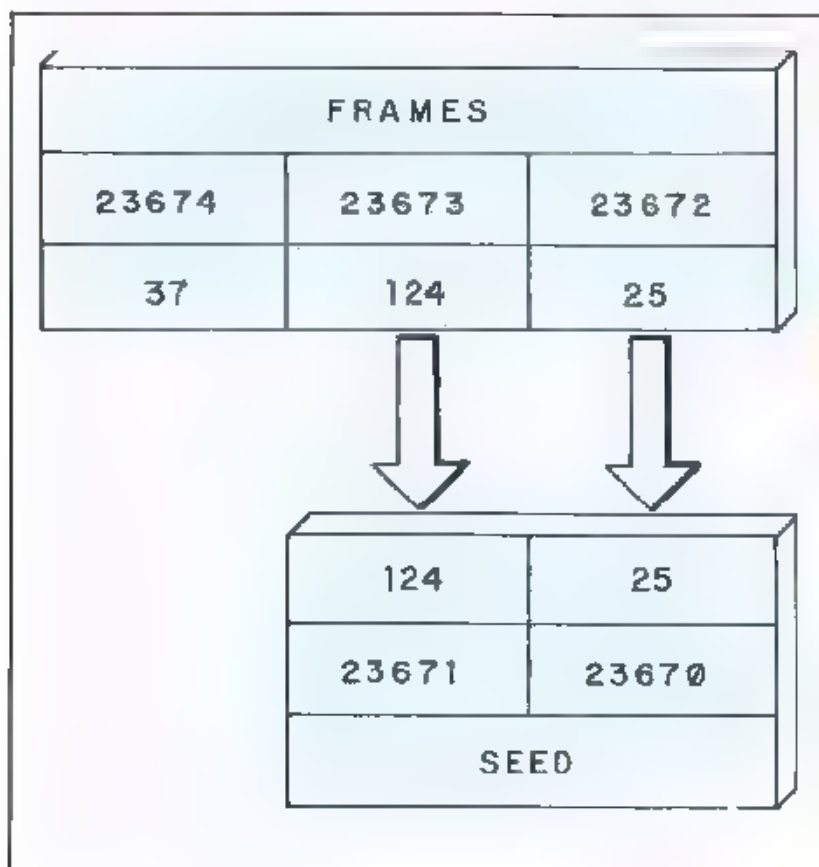
```
10 FOR n = 1 TO 44
20 PRINT RND
30 NEXT n
```

observará que todos los números son inferiores a «1», ya que el valor retornado por «RND» está comprendido entre «0» y «1», en alguna ocasión puede valer «0», pero nunca «1».

Para conseguir números aleatorios comprendidos entre otros rangos, por ejemplo,



Randomize «n».



Randomize «0».

de «0» a «10», podríamos utilizar:

```
10 FOR n = 1 TO 44
20 PRINT RAND * 10
30 NEXT n
```

pero si lo que se desea es obtener números aleatorios enteros, utilizaríamos:

```
10 FOR n = 1 TO 44
20 PRINT INT (RAND * 10)
30 NEXT n
```

Existe un pequeño algoritmo para obtener números aleatorios enteros comprendidos entre dos cualesquiera, ambos inclusive

$$INT ((Y - X + 1) * RAND) + X$$

donde «X» es el menor e «Y» el mayor, veamos unos ejemplos

```
10 LET y = INT (22 * RAND)
20 LET x = INT (32 * RAND)
30 PRINT AT y, x
40 GOTO 10
```

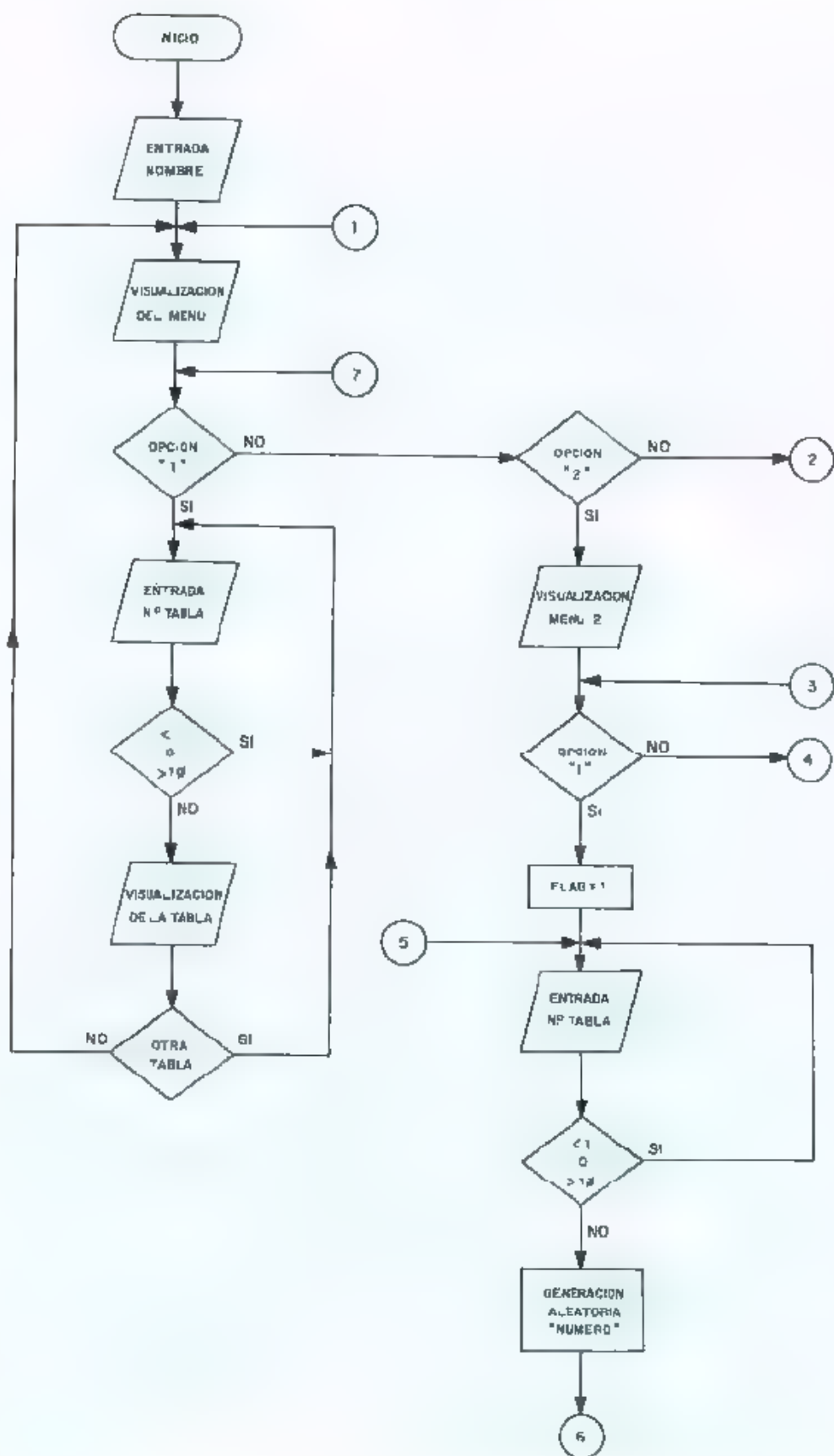
Este programa visualiza un asterisco (*) en una posición aleatoria de la pantalla si tiene paciencia, al cabo de un rato, verá la pantalla llena de este símbolo, en total «704».

Una variante de este programa podría consistir en introducir estas tres líneas suplementarias

```
5 INPUT Tiempo -> X
12 PAUSE X
34 PRINT AT y, x
```

la variable «X» controla el tiempo de visualización del astorico. Introduzca, por ejemplo, el valor «5» y parecerá que hay una mosca revoloteando sobre la pantalla de su televisor

Dentro del Sistema Operativo del Spectrum está implementado un algoritmo que calcula sucesivamente cual de los «65536» números aleatorios será el siguiente en ser presentado. El resultado de este algoritmo se almacena en dos posiciones consecutivas de memoria, éstas forman una de las variables del sistema conocida por el nombre de «SEED» el contenido de esta variable es, por



tanto, la dirección dentro de la tabla de números aleatorios donde se encuentra el siguiente a presentar

Edite las siguientes líneas de programa, al ejecutarlas, aparecerá en una columna el contenido de la variable SEED y en la otra el número aleatorio correspondiente.

La línea «50» se encarga de visualizar el contenido de dicha variable

El programa número «1» está basado en el conocido juego de los *barquitos*, pero en esta ocasión en vez de tener que hundir una completa flota enemiga compuesta por varias unidades, tan sólo será necesario acertar la posición de una lancha que está situada en una retícula de «10» por «10»

Las instrucciones del juego son sencillas, deberá introducir primero la coordenada vertical (y), posteriormente la horizontal (x). Cuando una de las coordenadas coincide aparece un mensaje de «alerta» indicando que vamos por buen camino.

La estructura general del programa es la siguiente.

10	: Presentación del programa
20	: Asignación de los colores azul para el borde, verde para el fondo y negro para los caracteres
30	: Inicialización de la variable «record»
60-140	Bucles utilizados para dibujar la retícula. Se han utilizado las sentencias «PLOT» y «DRAW» éstas serán explicadas en el capítulo dedicado al dibujo
150-180	Visualización de las posiciones de la retícula
220-250	Cálculo de la posición del barco. Se utiliza «RANDOMIZE», de esta forma, la posi-

ción depende del tiempo que lleve corriendo el programa

260	: Comienzo de juego
270-320	Inicialización y visualización de variables
330-400	: Introducción de valor de las coordenadas. Este tiene que ser un número entero y estar comprendido dentro de los márgenes (0 a 9)
410	: Tipo de la posición disparada
420-427	: Incremento y visualización de número de intentos
430	: Comprobación si las coordenadas han sido acertadas
440	: Comprobación si una de las coordenadas ha sido acertada
450-500	: Indicación de disparo fallido
510	: Peticion de nuevas coordenadas
520-600	: Indicación de «acierto» y visualización de la puntuación y actualización del record
670-730	: Continuación del juego?

RANDOMIZE

Acceso al teclado

RND



MERGE

MODO K

Tipo de sentencia

Comando de programación.

Definición

Si intenta buscar la palabra

clave «RANDOMIZE» no la encontrará, ya que en su lugar está «RAND», forma simplificada de la anterior

Su estructura general es

SENTENCIA	ARGUMENTO
RANDOMIZE	valor entero entre 0 y 65535

Ejemplos:

- RANDOMIZE 70
- RANDOMIZE date
- RANDOMIZE 10 * nlin/3
- RANDOMIZE

«RANDOMIZE» sin argumento toma, por defecto, el valor «0»

El argumento debe estar comprendido entre «0» y «65535», de lo contrario aparecerá el mensaje

Bad argument to RND

Existen dos formas de utilizar «RANDOMIZE»

a) Cuando su argumento tiene un valor comprendido entre «0» y «65535»

El argumento se utiliza para definir en qué número de la secuencia va a comenzar la generación de números aleatorios

Ejemplo:

```
10 RANDOMIZE 50
20 OR n = 10 * n
30 PRINT RND
40 NEXT n
```

Observará que cada vez que se ejecuta el programa la secuencia de número aleatorio es exactamente la misma, ya que siempre se inicializa con el valor «50».

Una forma de poder visualizar la lista completa de números aleatorios, podría ser la siguiente.

PROGRAMA 1

```

10 REM *****
*   CURSO BASIC   *
*****
*   !! AGUA !!   *
*****

20 BORDER 1 PAPER 4 INK 0 C
L5 30 LET record 0
60 REM *****
*   DISJUNTO   *
*****

70 FOR n=72 TO 152 STEP 8
80 PLOT 24,n

90 DRAW 80,0
100 NEXT n
110 FOR n=24 TO 104 STEP 8
120 PLOT n,72
130 DRAW 0,80
140 NEXT n
150 REM *****
*   COORDENADAS   *
*****

160 FOR n=3 TO 12
170 PRINT AT n,1,n-3
175 PRINT AT 1,n,n-3
180 NEXT n
220 REM *****
*   SITUACION   *
*****

30 RANDOMIZE
240 LET barcoy=INT (10*RND)
250 LET barcox=INT (10*RND)
260 REM *****
*   COMIENZO   *
*****

270 LET intentos=0
272 LET puntuacion=0
280 PRINT AT 3,16;"RECORD "
record
290 PRINT AT 5,16;"PUNTUACION "
puntuacion
300 PRINT AT 7,16;"INTENTOS "
intentos
310 PRINT AT 10,16;"COORD y "
320 PRINT AT 12,16;"COORD x "
330 REM *****
*   DIAPARA   *
*****

```

```

340 PRINT AT 10,25," ";AT 12,25
350 INPUT "Coordenada y >>> ",p
360 IF (INT posy<>posy) OR (pos
y<0) OR (posy>9) THEN GO TO 350
370 PRINT AT 10,25,posy
380 INPUT "Coordenada x >>> ",p
390 IF (INT posx<>posx) OR (pos
x<0) OR (posx>9) THEN GO TO 370
400 PRINT AT 12,25,posx
410 PRINT AT posy+3,posx+3,"■"
420 REM *****
*   COMPROBACION   *
*****

425 LET intentos=intentos+1
427 PRINT AT 7,27,intentos
430 IF (posy=barcoy) AND (posx=
barcox) THEN GO TO 520
440 IF (posy=barcoy) OR (posx=b
arcox) THEN PRINT FLASH 1,AT 14,
3;"ALERTA ";GO TO 460
450 PRINT AT 14,3,"
460 PRINT AT 18,10;"!!! AGUA !!"
490 PAUSE 50
500 PRINT AT 18,10,
510 GO TO 040
520 REM *****
*   ACIERTO   *
*****

530 PRINT AT 14,3,"
540 PRINT AT 18,9;"!!! ACERTO !"
550 PRINT AT posy+3,posx+3,FLASH
1,
560 IF intentos>100 THEN LET pu
ntuacion=0 GO TO 580
570 LET puntuacion=puntuacion+1
580 PRINT AT 5,27,puntuacion
590 IF puntuacion>record THEN L
ET record=puntuacion
600 PRINT AT 3,27;record
670 REM *****
*   CONTINUACION   *
*****

680 PRINT #0,AT 1,2,"Desea jugar
otra vez (S/N) "
690 PAUSE 0
700 LET a$=INKEY$
710 IF a$="S" OR a$="s" THEN CL
GO TO 60
720 IF a$="N" OR a$="n" THEN CL
STOP
730 GO TO 690

```

```

10 FOR n = 1 TO 85535
20 RANDOMIZE n
30 PRINT n, RND
40 NEXT n

```

Si tiene la suficiente paciencia podrá averiguar cual

es el número de secuencia que hace que la función «RND» retorne el valor «0», pero si no, compruébelo con

```

10 RANDOMIZE 45430
20 PRINT RND

```

Cuando se ejecuta una sentencia del tipo «RANDOMIZE n» lo que en realidad sucede es que la variable «SEED» asume el valor del argumento, el siguiente programa lo demuestra.

PROGRAMA 2

```

10 REM *****
   *          *
   *  CURSO BASIC  *
   *          *
   * ***** *
   *          *
   *   LA TABLA   *
   *          *
   * ***** *
20 BORDER 4  PAPER 4  INK 0  C
LS
99 REM
   *****
   *          *
   *  MENU  *
   *          *
   * ***** *

100 INPUT "Como te llamas >>>
' LINE 0$
110 PRINT AT 1,1,a$," elige una
de las tres"" opciones "
120 PRINT AT 9,8," 1 - REPASAR
130 PRINT AT 12,8," 2 - EXAMEN
132 PRINT AT 15,8," 3 - FIN
140 IF INKEY$="" THEN GO TO 140
150 IF INKEY$="1" THEN GO TO 10
00
160 IF INKEY$="2" THEN GO TO 20
00
162 IF INKEY$="3" THEN CLS : ST
OP
170 GO TO 140
1000 REM
   *****
   *          *
   *  REPASAR  *
   *          *
   * ***** *

1002 BORDER 1
1004 CLS
1010 INPUT "Que tabla deseas rep
asar >>> tabla
1020 IF tabla<1 OR tabla>10 THEN
GO TO 1010
1022 PRINT AT 1,2 "TABLA DE MULT
IPlicar de ",tabla,
1030 FOR n=1 TO 10
1040 LET resultado=tabla*n
1050 PRINT AT n+6,10,tabla, " x
",n,
1060 IF n< 10 THEN PRINT " ",
1070 PRINT " = ",resultado
1080 NEXT n
1090 PRINT AT 0 " Deseas repasar o
tra (S/N) "
1100 IF INKEY$="" THEN GO TO 110
0
1110 LET b$=INKEY$
1120 IF b$="S" OR b$="s" THEN GO
TO 1004
1130 IF b$="N" OR b$="n" THEN BO
RDER 4  CLS  GO TO 110
1140 GO TO 1100

2000 REM
   *****
   *          *
   *  MENU 2  *
   *          *
   * ***** *

2010 BORDER 2
2020 CLS
2022 FOR n=1 TO 25 NEXT n
2030 PRINT AT 1,1,a$," elige el
tipo "
2040 PRINT AT 8,8," 1 - PARCIAL "
2050 PRINT AT 12,8," 2 - GENERAL "
2060 IF INKEY$="" THEN GO TO 206
0

```

```

2070 LET b$=INKEY$
2080 IF b$="1" THEN LET flag=1
GO TO 2200
2090 IF b$="2" THEN LET flag=0
GO TO 2200
2100 GO TO 2070
2200 REM
   *****
   *          *
   *  PREGUNTAS  *
   *          *
   * ***** *

2210 CLS
2212 RANDOMIZE
2220 PRINT AT 2,1, PREGUNTAS
2230 PRINT AT 4,1, ACEPTOS
2240 PRINT AT 6,1, PORCENTAJE
2250 LET pregunta=1 LET acierto
0 LET porcentaje=0
2260 IF flag=0 THEN GO TO 2350
2270 INPUT " Que tabla deseas >>
" tabla
2280 IF tabla<1 OR tabla>10 THEN
GO TO 2270
2290 LET numero=INT (RND*10)+1
2300 GO TO 2400
2350 LET tabla=INT (RND*10)+1

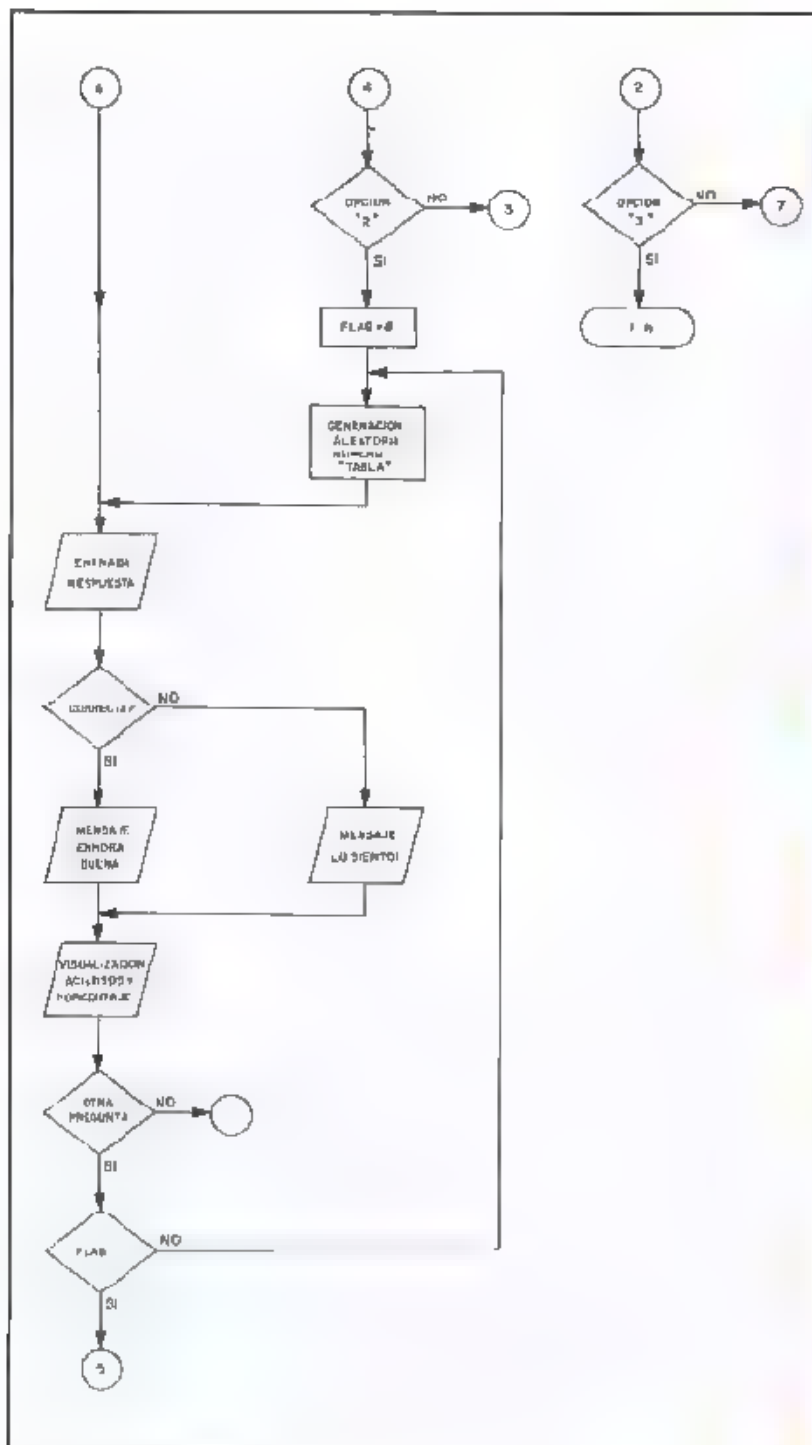
2350 LET numero=INT (RND*10)+1
2400 LET resultado=tabla*numero
2405 PRINT AT 13,pregunta,
2410 PRINT AT 10,1,tabla:" x " ,n
UMERO,
2420 INPUT " Cuntas son ", tab
la, " x ",numero," >>> " ;respuesta
2440 PRINT AT 10,9,respuesta
2450 REM
   *****
   *          *
   *  ANALISIS  *
   *          *
   * ***** *

2460 IF respuesta=resultado THEN
GO TO 2400
2470 GO TO 2510
2480 PRINT AT 15,7, " !! ENHORABU
ENA !! "
2490 LET acierto=acierto+1

2500 GO TO 2530
2510 PRINT AT 15,7," !! LO SIENT
O !! "
2520 PRINT AT 19,1,a$+" ",tabla,
" x " ,numero," son ",resultado
2530 PRINT AT 4,13,acierto
2540 LET porcentaje=INT (acierto
*pregunta*10000)/100
2550 PRINT AT 6,13,"
2560 PRINT AT 6,13,porcentaje,"
%"
2570 PRINT AT 0;" Deseas continuar
(S/N) "
2580 PAUSE 0
2590 LET b$=INKEY$
2600 IF b$="S" OR b$="s" THEN GO
TO 2530
2610 IF b$="N" OR b$="n" THEN BO
RDER 4  CLS  GO TO 110
2620 GO TO 2580
2630 LET pregunta=pregunta+1
2640 FOR n=15 TO 21

2650 PRINT AT n,0," "
2660 NEXT n
2670 IF flag=1 THEN GO TO 2290
2680 GO TO 2350

```



Programa «TABLA»

```

10 REM *****
11 VARIABLE FRAMES
12 *****
20 PRINT "Frames" : Seed
30 PRINT
40 RANDOMIZE
50 PRINT "PEEK 23674+65536+
K 23672 PEEK 23671+256+ PEEK
23670
60 GO TO 40
  
```

b) Cuando su argumento es cero.

Cada vez que se conecta el ordenador, se ejecuta una sentencia «NEW» o se activa el «RESET», la generación de números aleatorios se realiza siempre a partir del mismo valor

Edite el siguiente programa después de introducir «NEW» como comando directo

```

10 PRINT RND
20 GOTO 10
  
```

anote a uno de los valores obtenidos. Vuelva a introducir «NEW» y edite otra vez el programa. Comprobará que los resultados son idénticos.

Para conseguir una función más aleatoria debe introducirse previamente

```
RANDOMIZE 0
```

o en su defecto,

```
RANDOMIZE
```

Ejemplo

```

10 RANDOMIZE
20 PRINT RND
30 GO TO 20
  
```

En este caso el valor retornado por «RND» está en función del tiempo que lleva conectado el ordenador. Existe otra variable del sistema conocida por el nombre de «FRAMES» que almacena indirectamente este tipo. La variable «FRAMES» ocupa tres posiciones de memoria, veamos un ejemplo de lectura de dicha variable.

```

10 PRINT
PEEK 23674+65536+
PEEK 12+673+256+
PEEK 23672
20 GOTO 10
  
```

Cuando se ejecuta «RANDOMIZE 0» la variable «SEED» asume el valor de las dos posiciones de memoria menos significativas de «FRAMES». Veamos un ejemplo:

```

10 REM *****
   RANDOMIZE 5
*****
12 PRINT Randomize Seed
14 PRINT
20 INPUT Numero 1 a2a7
30 RANDOMIZE 4557
40 PRINT a2a7 PEEK 23570+PEEK
23570+50
50 GO TO 20

```

```

10 REM *****
   VARIABLE SEED
*****
12 LET Seed=23570
20 PRINT "Variable "SEED"
Nuestro aleatorio"
30 PRINT
40 FOR n=1 TO 10
50 PRINT PEEK Seed+PEEK ise
n*10+50
60 PRINT AND
70 NEXT n

```

Puede observar que, en ocasiones, existe una diferencia entre ambas, de una unidad debido al retardo que hay entre la ejecución de «RANDOMIZE» y la visualización de contenido de la variable «FRAMES»; recuerde que «FRAMES» se incrementa según transcurre el tiempo y «SEED» se actualiza al ejecutarse «RANDOMIZE».

El programa número «2» pertenece al grupo denominado «UTILIZADES» y tiene aplicación en el campo de la enseñanza, ya que permite a los principiantes estudiantes de la EGB repasar la tabla de multiplicar o contestar las preguntas que el ordenador responde sobre el tema.

Lo primero que hace el ordenador es preguntarnos nuestro nombre, ya que en diversas ocasiones hace referencia a él. Posteriormente

se presenta un menú con tres opciones

```

REPASAR
EXAMEN
FIN

```

La opción «1» permite repasar la tabla de multiplicar del número que elijamos, ya que se nos visualiza ésta en la pantalla.

La opción «EXAMEN» tiene otras dos opciones

```

1 PAR "A"
2 GEN "AL"

```

Eligiendo la primera, el ordenador nos pregunta la tabla de multiplicar de un número determinado, en cambio con la segunda, las preguntas son sobre cualquier número.

La opción «3» permite parar la ejecución del programa. La estructura general es,

```

10      - Comentario con el nombre de programa
20      - Asignación del color de pantalla fondo y borde, y negro para los caracteres.
100     - Entrada del nombre
110-132 - Presentación del menú
140-170 - Selección de la opción elegida
1000    - Comienzo de la opción (1) (REPASAR)
1002    - Borde de color azul
1010-1020 - Entrada y comprobación del número de tabla

```

```

1022-1080 Visualización de la tabla
1090-1140 Selección de una nueva tabla, o salto al menú principal
2000      Comienzo de la opción (2)
2010      Borde de color rojo
2322-2050 Presentación del menú secundario
2060-2100 Selección de la opción elegida
2200      Comienzo de las preguntas
2212      Inicialización de la selección de números a adivinar, esta depende de tiempo que lleva conectada el ordenador
2220-2250 Visualización de algunas e inicialización de variables
2260-2360 Generación aleatoria de la pregunta, dependiendo de la opción elegida
2400      Cálculo del resultado
2405-2420 Visualización de la pregunta y entrada de la respuesta.
2440      Visualización de la respuesta
2450      Comienzo de análisis
2460      Comparación de los resultados
2480-2490 Mensaje de acierto
2510-2520 Mensaje de fallo
2530-2560 Visualización del número de aciertos y porcentaje
2570-2620 Selección de otra pregunta o salto al menú principal
2630      Incremento de la variable «pregunta»
2640-2660 Borrado de mensaje
2670-2680 Selección de la línea de salto para una nueva pregunta.

```


FUNCIONES DE CADENA

En capítulos anteriores se explicaron las funciones matemáticas y la pseudotomática; en éste, por el contrario, van a ser estudiadas las *funciones de cadena*.

Conviene, antes de leer este capítulo, si no se tiene el concepto de *cadena* lo suficientemente claro, repasar el dedicado al «Código ASCII» (pag 37) y «operaciones con

cadenas» (pag. 42), en este último se tratan temas tales como:

- CONCATENACION
- SUBCADENAS
- FRAGMENTACION
- ORDENACION

Las funciones de cadena implementadas dentro del juego de sentencias del

Spectrum son:

- LEN
- STR\$
- VAL
- VAL\$
- CHR\$
- CODE

también pueden definirse funciones de cadena por el usuario con la sentencia «DEF FN»

PROGRAMA 1

```

10 REM *****
   *      *
   *  CURSO/BASIC  *
   *      *
   * *****      *
   *      *
   *  BUSQUEDA      *
   *      *
   * *****      *
   *      *

20 PRINT "Introduce una cadena
max=30"
30 INPUT "> ", LINE a$
40 LET longitud=LEN a$
50 IF longitud<1 OR longitud>3
 THEN GO TO 30
60 CLS
70 PRINT "CADENA."
80 PRINT a$
90 PRINT "LONGITUD= ", longitud
100 PRINT "CARACTER A BUSCAR."

110 INPUT "> ", LINE b$
120 IF b$="" THEN GO TO 110
130 LET b$=b$ 1)
140 PRINT b$
150 LET contador=0
160 FOR x=1 TO longitud
170 LET c$=a$(x)
180 IF c$=b$ THEN PRINT c$, "
   GO TO 205
190 LET contador=contador+1
200 PRINT "FALSA 1 c$
205 PAUSE 10
210 NEXT x
220 PRINT
230 IF contador=0 THEN PRINT "C
   aracter no encontrado." GO TO 2
240 PRINT "Caracter encontrado
   " contador " veces."
250 PRINT "¿Que quiere buscar oir
   o caracter S/N"

260 PAUSE 0
270 LET i$=INKEY$
280 IF i$=S OR i$="s" THEN CL
   S GO TO 70
290 IF i$="N" OR i$="n" THEN ST
   OP
300 GO TO 260

```

PROGRAMA 2

```

10 REM *****
   *      *
   *  CURSO/BASIC  *
   *      *
   * *****      *
   *      *
   *  INSERTAR      *
   *      *
   * *****      *
   *      *

20 PRINT "Introduce una cadena
(max=30)"
30 INPUT "> ", LINE a$
40 LET longitud=LEN a$
50 IF longitud<1 OR longitud>3
 THEN GO TO 30
60 CLS
70 PRINT "CADENA "
80 PRINT a$
100 PRINT "CADENA A INSERTAR "
110 INPUT "> ", LINE b$

120 IF b$="" THEN GO TO 110
130 IF LEN b$>30 THEN GO TO 110
140 PRINT b$
150 PRINT "POSICION "
160 INPUT "> ", LINE n$
170 LET posicion=VAL n$
180 IF posicion<1 OR posicion>l
ongitud THEN GO TO 160
185 PRINT posicion
190 LET c$=a$(1 TO posicion)
200 LET fin=longitud-posicion
210 IF fin=0 THEN LET d$="" GO
   TO 230
220 LET d$=a$(posicion+1 TO )
230 LET a$=c$+b$+d$
240 PRINT a$
250 FOR x=1 TO LEN b$
260 PRINT b$(x),
265 PAUSE 10
270 NEXT x
280 PRINT d$
285 LET longitud=LEN a$

285 LET longitud=LEN a$
290 PRINT "Mas Caracteres a
   Insertar S/N"
300 PAUSE 0 LET i$=INKEY$
310 IF i$="S" OR i$="s" THEN GO
   TO 60
320 IF i$="N" OR i$="n" THEN ST
   OP
330 GO TO 300

```

Acceso al teclado

LEN



MODO E

SCREEN 3

Definición

La función «LEN» retorna un valor equivalente al número de caracteres de una cadena, hay que tener presente que los espacios en blanco también cuentan

La estructura general de esta sentencia es

FUNCION	ARGUMENTO
LEN	expresión de cadena

Ejemplos.

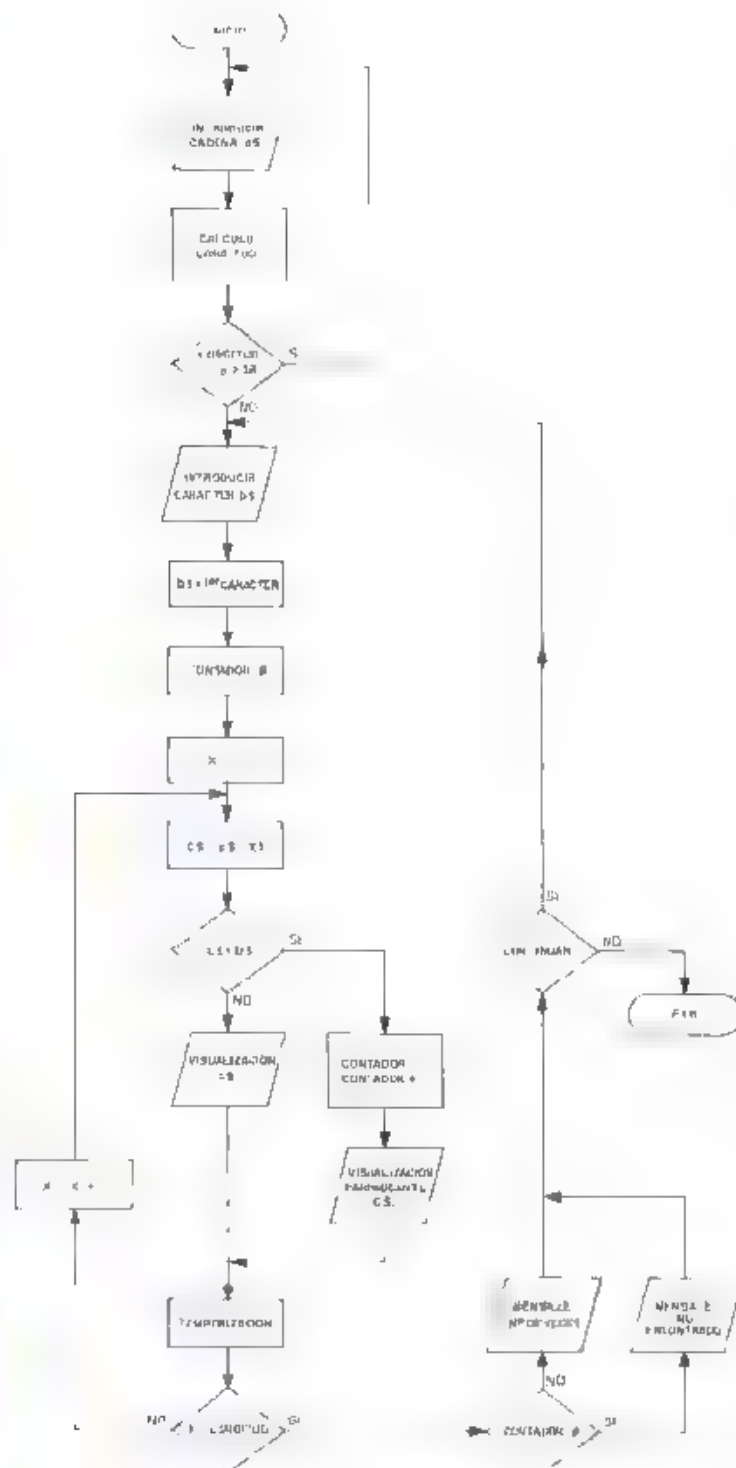
- PRINT LEN "BASIC"
- PRINT LEN a\$
- LET C\$ = LEN (b\$ + j\$)
- LET X\$ = LEN a\$ (3 TO)

El siguiente programa es un ejemplo de utilización de la sentencia «LEN»:

```

10 REM *****
20 FUNCTION LEN
30 PRINT "Introduce la cadena"
40 INPUT a$
50 FOR i=1 TO LEN(a$)
60 IF a$(i) = " " THEN LET i=i+1
70 NEXT i
80 PRINT "La longitud es: "; i
90 RETURN i
100 PRINT "Fin"
110 END

```



Estructura programa «BUSQUEDA».

este, calcula la cantidad total de caracteres que forman el nombre introducido por el teclado. El bucle formado por las líneas «50» a «70» detecta los espacios en blanco y son restados de la longitud total.

Cuando se evalúa una cadena vacía, el valor retornado es igual a «0».

STR\$

Acceso al teclado

STR\$



[

MODO [

Definición

Asigna a una variable de cadena el resultado de una expresión numérica. Su estructura general es:

FUNCIÓN	ARGUMENTO
STR\$	expresión numérica

Ejemplos:

- PRINT STR\$ 7
- PRINT STR\$ X
- LET n = STR\$ (7 * 12)
- LET x = STR\$ (v n)

Veamos el siguiente programa

```
10 REM *****
  FUNCION STR$
  *****
```

```
20 INPUT "¿A qué edad eres?"
30 LET b$=STR$(NUM$ 0)
40 LET b$=VAL$(INTRODUCIDO)

45 CLS
50 PRINT "b$=" & b$
60 GO TO 20
```

En la línea «30» se asigna a la variable de cadena «b\$» el valor de la variable numérica «numero», de esta manera puede realizarse la concatenación de la línea «50»

VAL

Acceso al teclado

VAL



VAL \$

MODO [

PROGRAMA 3

```
10 REM *****
  * CURSO/BASIC *
  * ***** *
  * ANULAR *
  * ***** *

20 PRINT "Introduce una cadena"
  (max. 30)"
30 INPUT "> ", LINE a$
40 LET longitud=LEN a$
50 IF longitud<1 OR longitud>30
  THEN GO TO 30
60 CLS
70 PRINT CADEM$
80 PRINT a$
150 PRINT "POSICION CARACTER A"
  BORRAR
160 INPUT "> ", LINE n$
```

```
170 LET posicion=VAL n$
180 IF posicion<1 OR posicion>
  longitud THEN GO TO 160
190 PRINT posicion
200 LET c$=a$(1 TO posicion)
210 LET fin=longitud-posicion
220 IF fin=0 THEN LET d$="" GO
  TO 230
230 LET d$=a$(posicion+1 TO )
240 LET a$=c$+d$
250 PRINT a$
260 LET longitud=LEN a$
270 PRINT "Mas caracteres a"
  borrar S/N"
300 PAUSE 0 LET i$=INKEY$
310 IF i$=0 OR i$=S THEN GO
  TO 60
320 IF i$="N" OR i$="n" THEN ST
  OP
330 GO TO 300
```

CADENA:
MICROHOBBY SEMANAL

LONGITUD: 10

CARACTER A BUSCAR: M

MICROHOBBY SEMANAL

Caracter encontrado 2 veces.

CADENA:

MICROHY SEMANAL

CADENA A INSENIAR

088

POSICION: 6

MICROHOBBY SEMANAL

Búsqueda de caracteres.

Definición

Evalúa una cadena cuyo contenido son números y operadores matemáticos, y el resultado lo asigna a una variable numérica.

Su estructura general es:

FUNCION	ARGUMENTO
VAL	expresión de cadena

Ejemplos.

- PRINT VAL "23"
- LET a = VAL a\$
- PRINT VAL ("24" + "/"2")
- LET n = VAL b\$ + c\$

Para realizar la evaluación, la función «VAL» suprime las comillas de la cadena

El siguiente programa realiza las cuatro operaciones básicas entre dos números, estos están asignados a las variables «a\$» y «b\$». El signo de la operación depende de la letra minúscula introducida por teclado (s – suma, r – resta, m – multiplicación y d – división) y queda asignada a la variable «C\$».

Para calcular el resultado se utiliza la función «VAL» en la línea 100.

Inserción de cadenas.

```
10 REM *****
11 * FUNCION VAL *
12 * *****
20 INPUT "Numero 1 >>> ",a$
30 CLS
40 INPUT "Numero 2 >>> ",b$
45 INPUT "Operacion (s/r/m/d) >>> ",c$
50 IF c$="s" THEN LET c$="+"
60 IF c$="r" THEN LET c$="-"
70 IF c$="m" THEN LET c$="*"
80 IF c$="d" THEN LET c$="/"
90 GO TO 100
100 LET resultado=VAL (a$+c$+b$)
110 PRINT a$, " ", c$, " ", b$, " = ", resultado
120 GO TO 20
```

Ejemplos:

- PRINT VAL\$ ""BASIC""
- LET a\$ = VAL\$ "b\$"
- PRINT VAL\$ "a\$ + c\$"
- LET b\$ = VAL\$ "C\$" + "PE-PE"

Conversiones de código

Hay dos funciones de cadena que permiten realizar cualquier conversión entre los códigos «ASCII» y «DECIMAL», éstas son:

VAL\$

Acceso al teclado

VAL



VAL\$ Definición

La función «VAL\$» evalúa variables, constantes o expresiones de cadena siempre que estén encerradas entre paréntesis; el valor retornado es otra cadena.

Su estructura general es:

FUNCION	ARGUMENTO
VAL\$	(expresión de cadena)

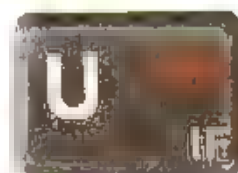
CHR\$
CODE

Ver tabla del Código ASCII en la pag. 41

CHR\$

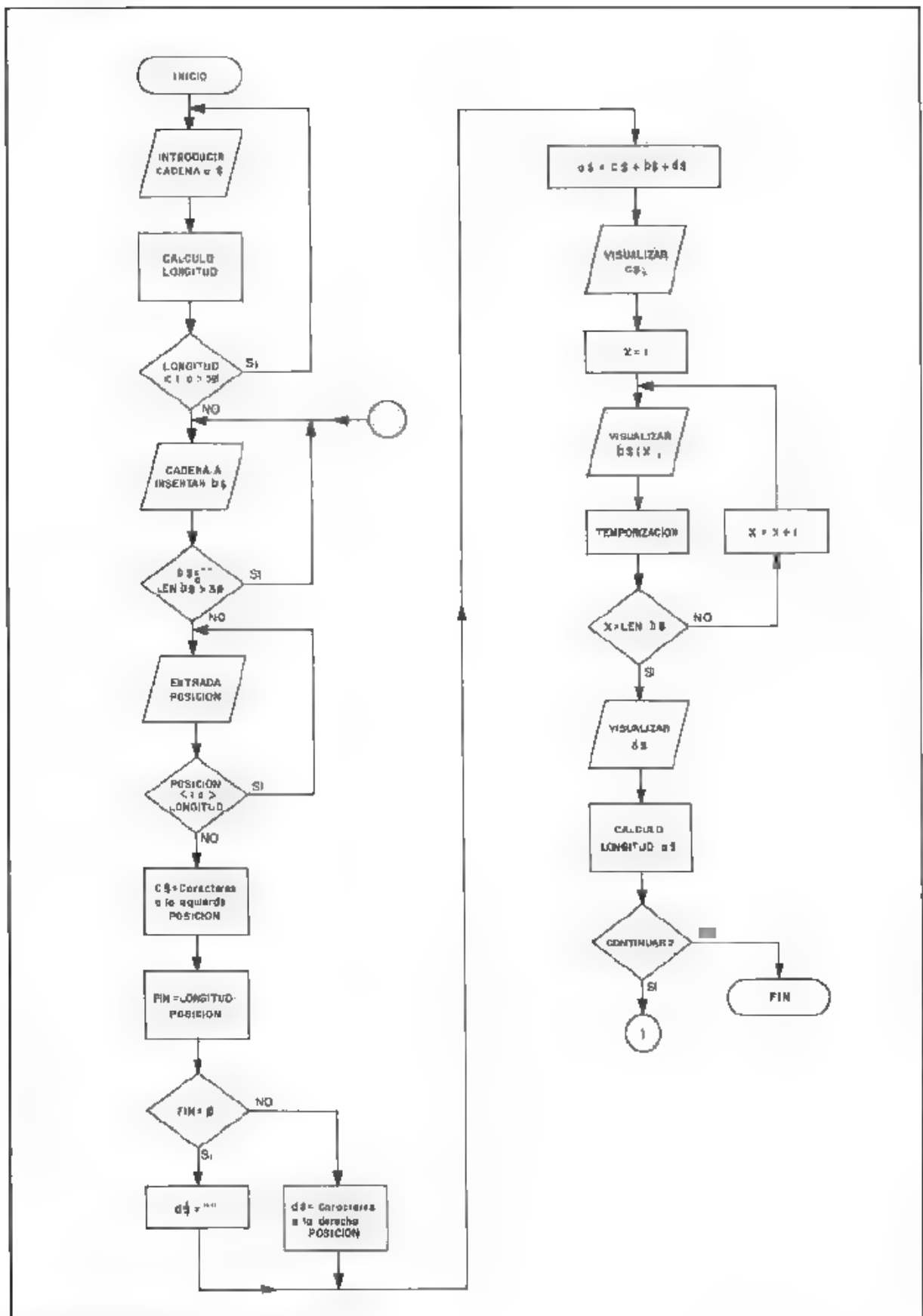
Acceso al teclado

CHR\$



Definición

La función «CHR\$» transforma un número decimal en



Estructura programa «INSERTAR».

FUNCIÓN	ARGUMENTO
CODE	expresión de cadena

Ejemplos.

- PRINT CODE «A»
LET n = CODE a\$
- PRINT CODE b\$ (X)
LET X 3 CODE «RAIZ»

Al retornar «CODE» el código del primer carácter, los siguientes comandos directos darán el mismo resultado (65):

```
PRN CODE "A"
PRINT CODE "A-baz!"
PRINT CODE "A mer a"
```

La palabra clave «CODE» también es utilizada junto con «LOAD», «SAVE» y «VERIFY», pero esta particularidad será vista en otro capítulo.

En este programa se visualizan las letras mayúsculas de la «A» a la «Z» con su correspondiente código decimal. Observe que la función «CODE» se utiliza para controlar el bucle.

```
10 FOR n = CODE "a" TO CODE "z"
20 PRINT : CHR$(n)
30 NEXT n
```

En este otro, debe introducirse una cadena como máximo de «15» caracteres; la instrucción 70 permite centrar la visualización de dicha cadena. El bucle compuesto por las líneas «80» a «100» permite representar el código decimal de cada carácter, ya que se utiliza la función «CODE» junto con la fragmentación de la cadena.

Cuando el argumento es una cadena vacía («») la función «CODE» retorna el valor «0».

```
10 REM *****
20 FUNCTION CODE$(S)
30 IF LEN(S) = 0 THEN CODE$ = 0
40 LET n = 1
50 IF LEN(S) > 15 THEN n = 15
60 DO WHILE n <= LEN(S)
70 PRINT AT 8, 10+ (n-1)*2
80 FOR a = 1 TO CODE$(S:n)
90 PRINT : a$ = CHR$(a)
100 NEXT a
110 n = n + 1
120 DO TO 20
```

Funciones definidas de cadena

De la misma manera que el usuario puede definir sus propias funciones numéricas, también puede hacerlo con las de cadena.

La estructura general es:

SENTENCIA	ARGUMENTO
DEF FN	para \$ (variables = función)

Ejemplos

```
DEF FN a$(b) = CHR$(b + 1)
```

- DEF FN b\$(a\$, c\$) = a\$ + c\$
- DEF FN c\$(a\$, n) = a\$(n)
- DEF FN d\$(X) = STR\$(X)

La llamada de las funciones sigue la estructura:

```
FN rutina$(parámetros)
```

Ejemplos:

- PRINT FN a\$(59),
- PRINT FN b\$ ("Pepe", "Juan")
- PRINT FN c\$ ("Camión", 3)
- PRINT FN d\$(4 + 56)

Errores

Dos errores típicos en el

manejo de cadenas son:

a) C Nonnense in BASIC

Ocurre en las siguientes situaciones.

```
IF a$ = "1374a2"
PRINT VAL a$
```

ya que la variable «a\$» contiene un carácter no numérico. Sin embargo

```
LET a$ = "a13742"
PRINT VAL a$
```

da el error

```
2 variable not found.
```

ya que la función «VAL» al suprimir las comillas y detectar que el primer carácter es una letra, el sistema operativo interpreta que se trata de una variable numérica y al no encontrarla visualiza el error anterior. Observe la diferencia con las siguientes instrucciones

```
LET valor = 30
IF a$ = "valor"
PRINT VAL a$
```

Con la función «VAL\$» también se produce el error «C».

```
LET c$ = "asis"
PRINT VAL$ c$
```

ya que o bien se modifica:

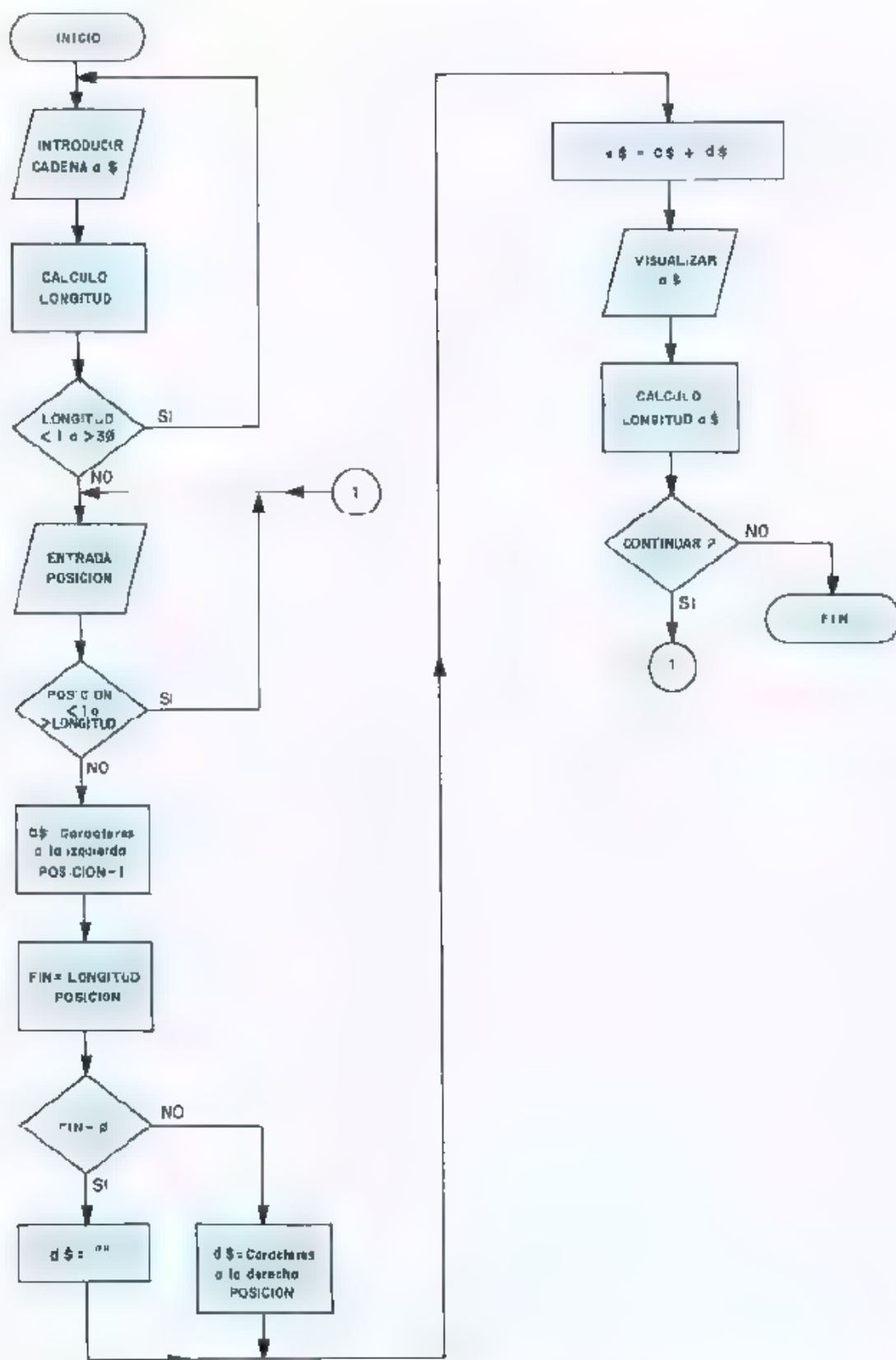
```
LET c$ = ""asis""
```

o por el contrario

```
PRINT VAL$ "C$"
```

b) 3 Subscript wrong

Sucede cuando el índice



Estructura programa «ANULAR».

de una fragmentación tiene un valor mayor que la longitud total de la cadena.

Ejemplo:

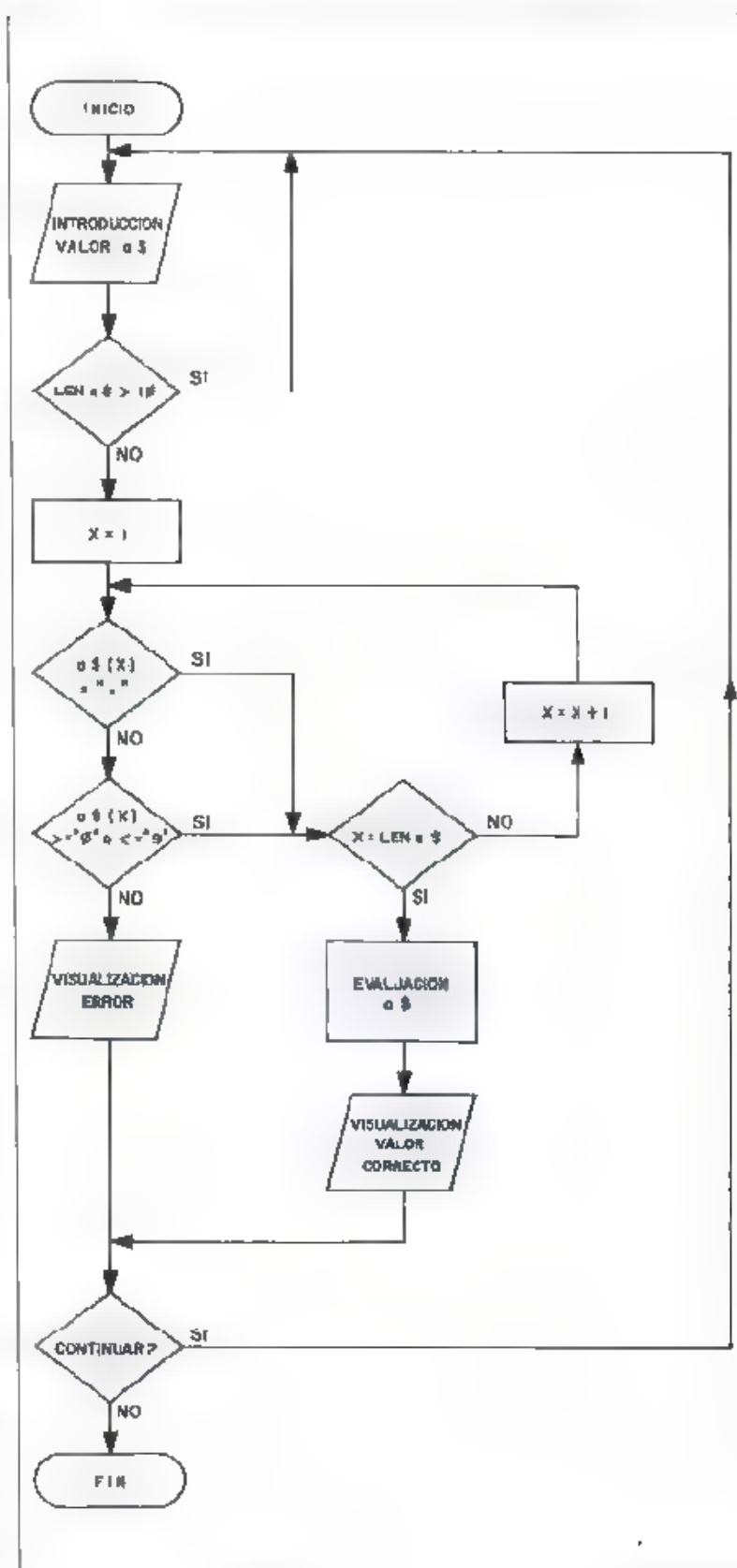
```
LET a$ = "sandra"
PRINT a$ (7)
```

Programa

El programa n.º 177 (BUSQUEDA) permite introducir una cadena con un máximo de 30 caracteres y, posteriormente, buscarnos un carácter determinado, si lo encuentra éste empezará a parpadear.

La estructura general es:

- | | | |
|---------|---|---|
| 10 | : | nombre del programa |
| 20 | : | Mensaje explicativo. |
| 30 | : | Entrada de la cadena (a\$) |
| 40 | : | Cálculo de la longitud de la cadena. |
| 50 | : | Comprobación de la longitud. |
| 60 | : | Borrado pantalla |
| 70 90 | : | Visualización de la cadena y su longitud |
| 100 110 | : | Entrada del carácter a buscar (b\$) |
| 120 | : | Comprobación de que no es una cadena vacía. |
| 130 | : | Asignación a la cadena b\$ de su primer carácter |
| 140 | : | Visualización del carácter a buscar. |
| 150 | : | inicialización de la variable contador a cero |
| 160 | : | Comienzo del bucle de búsqueda |
| 170 | : | Asignación a la variable C\$ del carácter muestreado. |
| 180 | : | Verificación si el carácter muestreado es igual al buscado: si no lo es, se visualiza el carácter |
| 190 200 | : | Si el carácter es igual se incrementa la variable |



Estructura programa «INPUT».

	contador en una unidad y se visualiza parpadeante (FLASH) el caracter	210	Si no hay ninguno, asigna a la variable d\$ una cadena vacía
205	- Temporización	220	Si hay, asigna a d\$ los caracteres situados a la derecha de posición
210	- Incrementa de la variable de control del bucle	230	- Concatenación de las variables c\$, b\$ y d\$
230	- Si la variable (contador) es igual a cero se visualiza el mensaje «Carácter no encontrado»	240	- Visualización de c\$
240	- Visualización de contenido de la variable (contador) en el caso de que sea distinto de cero	250 270	- Visualización, a continuación de los caracteres, de la variable b\$ (temporización)
250 300	- Analisis de las teclas (S) y (N) para buscar otro caracter o no.	280	- Visualización de la variable d\$
		285	- Nuevo cálculo de la longitud de la cadena
		290 330	- Analisis de las teclas (S) y (N) para insertar otro caracter o no.

El programa n.º 2 (INSERTAR) inserta un caracter o una serie de ellos dentro de otra cadena a partir de la posición que se quiera

La estructura general del programa es

10	Comenzar con el nombre del programa
20 30	Entrada de la cadena a\$
40	Cálculo de su longitud
50	- Verifica si está dentro de los límites
70 80	Visualización de la cadena
100 110	Entrada de la cadena a insertar (b\$)
120	- Comprueba que b\$ no es una cadena vacía
130	- Comprueba que no tiene mas de treinta caracteres
140	- Visualización de la cadena a\$
150 170	Entrada de la posición
180	- Comprueba que la posición no es cero ni mayor que la longitud de la cadena a\$
185	- Visualización de la posición
190	- Asignación a la variable c\$ de los caracteres situados a la izquierda de posición
200	- Cálculo cuántos caracteres hay situados a la derecha de posición

El programa n.º 3 (ANULAR) borra el caracter indicado por su posición, dentro de una cadena que tenga «30» caracteres como máximo.

La estructura de programa es

10	- Nombre del programa
20 30	- Entrada de la cadena a\$
40	- Cálculo de su longitud
50	- Comprueba si está dentro de los límites (1-30)
60	- Borrado de la pantalla
70 80	- Visualización de la cadena a\$
150 170	- Entrada de la posición a borrar
180	- Comprueba que no se encuentre situada fuera de la cadena
185	- Visualización de la posición
190	- Asignación de los caracteres situados a la izquierda de posición a la variable c\$
200	- Cálculo si el caracter a borrar es el último
210	- Si o es, asigna a la variable d\$ el valor de una cadena vacía
220	- Si no es, asigna a d\$ los caracteres situados a la derecha de posición

230	- Asigna a la variable a\$ el nuevo valor (c\$ + d\$ sin el caracter borrado)
240	- Visualización de la variable a\$
280	- Activación de la temporización de a\$
290 330	- Analisis de las teclas (S) y (N) para equilar un nuevo caracter o no.

El programa n.º (INPUT) detecta si dentro de una sentencia «INPUT» se ha introducido un caracter numérico, de esta manera se consigue que por error no aparezca el conocido mensaje

2 Variable, not found

La estructura del programa es.

10	Comenzar con el nombre del programa
20	Entrada de valor numérico asociado a una variable de cadena (a\$)
30	Comprobación de que la longitud no excede de 10 caracteres
40	Comienzo de bucle de análisis
50	- Si el código del caracter es un número continúa el analisis de caracteres
60	- Si el código pertenece a un rango de longitud entre (0) y (9) se continúa con el analisis
70	- El caracter al que se ha referido no un número hace que a ejecución de bucle se reanicie
80	- Incremento de la variable de control del bucle
90 100	- Si todos los caracteres son conocidos se evalúa la variable (valor) y se visualiza su contenido
120 130	- Visualización de un mensaje de error y del primer caracter no numérico introducido en la cadena
140 190	- Analisis de las teclas (S) y (N) para introducir un nuevo valor o no.

MATRICES

La matriz, tabla o array, es una estructura utilizada en programación que permite almacenar los datos (constantes o variables) de un programa.

Los elementos de una matriz tienen en común el mismo nombre y se diferencian en el subíndice, de esta manera se tiene el acceso directo a ellos.

Utilizando correctamente los subíndices pueden asignarse o leerse los contenidos de los elementos de una matriz.

Las matrices pueden ser de dos tipos.

NUMERICAS
DE CAJERA

Dimensionado de matrices

Para reservar una serie de posiciones de memoria (tabla) es necesario, previamente, *dimensionar* una matriz; tarea que consiste en definir la cantidad de elementos de que va a constar así como su distribución; es decir, podríamos dimensionar una matriz de 40 elementos, de varias formas:

- Una, en que los elementos están consecutivos.
- También podrían estar distribuidos como en una especie de «tablero de ajedrez», que tuviera «5» filas por «8» columnas o «10» filas por «4»

columnas, etc.

- Otra solución sería distribuirlos en dos planos formados cada uno por dos «tablas» de 4 * 5
- etc.

Dependiendo de la distribución las matrices, reciben diversos nombres:

UNIDIMENSIONALES
- BIDIMENSIONALES
- TRIDIMENSIONALES
MULTIDIMENSIONALES

ADVERTENCIA

Las matrices antes de ser utilizadas deben estar «dimensionadas».

PROGRAMA 1

```
10 REM *****
  * MANEJO DE TABLAS *
  *****

20 INPUT "Número de elementos"
>>> elementos
30 IF elementos<1 THEN elementos=1
40 IF elementos>100 THEN elementos=100
50 DIM a(elementos)
60 REM *****
  * ASIGNACION VALORES *
  *****

65 RANDOMIZE
70 FOR n=1 TO elementos
  LET a(n)=INT(RND*100)+1
80 NEXT n
90 REM
```

```
*****
* VISUALIZACION *
*****

120 FOR n=1 TO elementos
130 PRINT n,
132 IF n<10 THEN PRINT " ",
134 PRINT "> ",a(n),
140 NEXT n
150 REM *****
  * BUSQUEDA *
  *****

160 INPUT "Valor a encontrar" >
> valor
170 IF valor<1 THEN valor=1
180 IF valor>100 THEN valor=100
190 LET veces=0
200 DIM b(elementos)
210 REM
```

```

*****
*
* EXPLORACION *
*
*****

220 FOR n=1 TO elementos
230 IF a(n) <> valor THEN GO TO 2
60
240 LET veces=veces+1
250 LET b(veces)=n
260 NEXT n
270 REM

*****
*
* VISUALIZACION *
*
*****

280 CLS
290 IF veces=0 THEN PRINT "Valor
valor," no encontrado": GO TO 350
290 PRINT "El valor ",valor
300 PRINT "ha sido encontrado
veces," veces"
310 PRINT "en las posiciones:"

320 FOR n=1 TO veces
330 PRINT "> ",b(n)
340 NEXT n
350 PRINT #0,"Otro valor (S/N)"
360 PAUSE 0
370 LET z$=INKEY$
380 IF z$="S" OR z$="s" THEN GO
TO 150
390 IF z$="N" OR z$="n" THEN GO
TO 410
400 GO TO 350
410 CLS
412 FOR n=1 TO elementos
414 PRINT n,
416 IF n<10 THEN PRINT " ",
420 PRINT "> ",a(n),
430 NEXT n
440 REM

*****
*
* PAR/IMPAR *
*
*****

450 PRINT #0,"Pulsa una tecla p
ara continuar"
460 PAUSE 0
470 CLS
480 LET par=0
490 LET impar=0
500 DIM l(elementos)
510 DIM d(elementos)
520 REM

*****
*
* BUSQUEDA *
*
*****

530 FOR n=1 TO elementos
540 LET division=a(n)/2
550 IF INT (division)*2=a(n) TH
EN GO TO 590
560 LET impar=impar+1
570 LET d(impar)=a(n)

```

```

580 GO TO 610
590 LET par=par+1
600 LET c(par)=a(n)
610 NEXT n
620 REM

*****
*
* VISUALIZACION *
*
*****

625 CLS
630 PRINT "Numeros pares encont
rados ",par
640 FOR n=1 TO par
650 PRINT " ",c(n),
660 NEXT n
670 PRINT #0,"Pulsa una tecla p
ara continuar"
680 PAUSE 0
685 CLS
690 PRINT "Numeros impares enco
ntrados ",impar
700 FOR n=1 TO impar
710 PRINT " ",d(n),
720 NEXT n
730 PRINT #0;"Vuelvo a visualiz
ar los (S/N)"
740 PAUSE 0
750 LET z$=INKEY$
760 IF z$="S" OR z$="s" THEN GO
TO 620
770 IF z$="N" OR z$="n" THEN GO
TO 790
780 GO TO 740
790 CLS
800 FOR n=1 TO elementos
802 PRINT n,
804 IF n<10 THEN PRINT " ",
810 PRINT "> ",a(n),
820 NEXT n
830 REM

*****
*
* ORDENA *
*
*****

840 PRINT #0;"Pulsa una tecla p
ara continuar"
850 PAUSE 0
860 CLS
870 FOR x=1 TO elementos 1
880 LET menor=x
890 FOR y=x+1 TO elementos
900 IF a(menor)>a(y) THEN LET m
enor=y
910 NEXT y
920 LET cambio=a(x)
930 LET a(x)=a(menor)
940 LET a(menor)=cambio
950 PRINT x,
960 IF x<10 THEN PRINT " ",
970 PRINT "> ",a(x),
980 NEXT x
990 PRINT elementos,
1000 IF elementos<10 THEN PRINT
" ",
1010 PRINT "> ",a(elementos)
1020 PAUSE 0

```

DIM

Acceso al teclado

DATA



MODOS K

Tipo de sentencia

Comando de programa
ción.

Definición

La sentencia «DIM» permite
dimensionar tanto las matri-
ces de tipo numérico como
las de cadena.

Matrices numéricas

La estructura del dimensio-
nado de matrices numéricas
es.

SENTENCIA	ARGUMENTO
DIM	letra (lista de valores)

Ejemplos.

a) Matriz «unidimensional».

DIM X (20)

La matriz «X» se dimensiona de 20 elementos consecutivos. El subíndice varía, por tanto, entre 1 y 20.

X (1), X (2), X (20)

b) Matriz «bidimensional».

DIM J (5, 7)

La matriz «J» se dimensiona de 35 elementos distribuidos en 5 filas por 7 columnas. Los subíndices varían desde:

J (1, 1), J (1, 2), J (5, 6), J (5, 7)

c) Matriz «tridimensional».

DIM N (3, 4, 2)

Matriz llamada «N» constituida por 24 elementos distribuidos en tres planos de 4 filas por 2 columnas. Los subíndices varían entre:

N (1, 1, 1), N (1, 1, 2), N (3, 4, 2)

Los subíndices también pueden estar constituidos por variables y expresiones de tipo numérico.

Ejemplos:

- DIM C (7 * 3, 6)
- DIM P (Valor)
- DIM F (SIN cr, 5)
- DIM J (20 * X, SQR 9)

Como habrá podido observar, el nombre de una matriz está formado por una sola letra que la distingue de las demás, por tanto, no puede haber dos matrices con el mismo nombre aunque estén dimensionadas de distinta forma, ya que al dimensionar una matriz se borra cualquier otra que tuviera el mismo

MATRIZ "B".	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	

SUBINDICE

Matriz unidimensional B(18), elemento B(10).

nombre. Por el contrario, puede haber una variable numérica con el mismo nombre que una matriz, ya que, el sistema operativo distingue una de la

otra, por que, en su caso se utilizan los subíndices y en el otro no.

Cuando se dimensiona una matriz numérica el contenido

de sus elementos queda inicializado el valor cero

Ejemplo

```
10 DIM X(2)
20 FOR n = 1 TO 2
30 PRINT n * X(n)
40 NEXT n
```

Asignación y visualización

Para asignar un valor a un elemento determinado de una matriz, es necesario *posicionar* correctamente los subíndices o punteros. Por ejemplo:

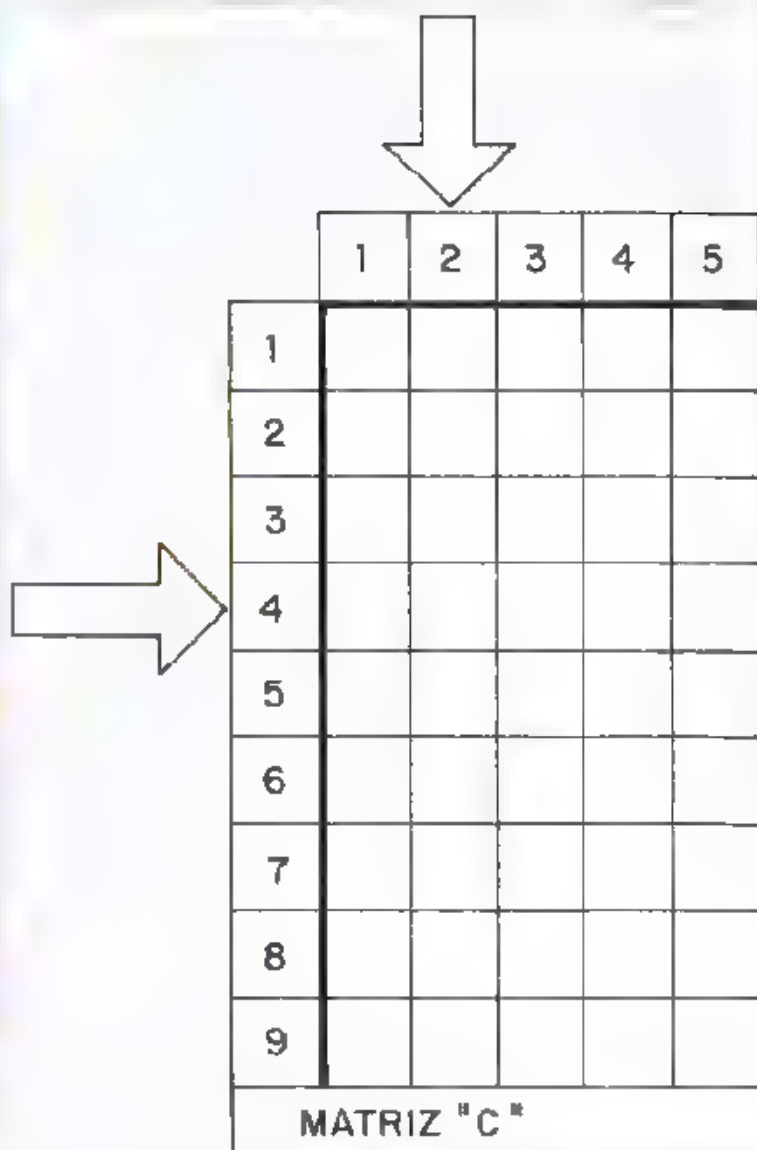
a) Asignar el valor 30 al elemento 7 de una matriz «Z»

```
LET Z(7) = 30
```

b) Asignar al elemento situado en las coordenadas «y = 3» y «x = 5» de una matriz «J», el valor de la variable «patron»

```
LET J,3,5 = patron
```

Hay que tener presente que los punteros deben estar comprendidos dentro de los valores especificados al dimensionar la matriz



Matriz bidimensional C(9,5), elemento C(4,2).

```
1 > 53
3 > 20
5 > 31
7 > 81
9 > 56
11 > 10
13 > 34
15 > 34
17 > 42
19 > 9
```

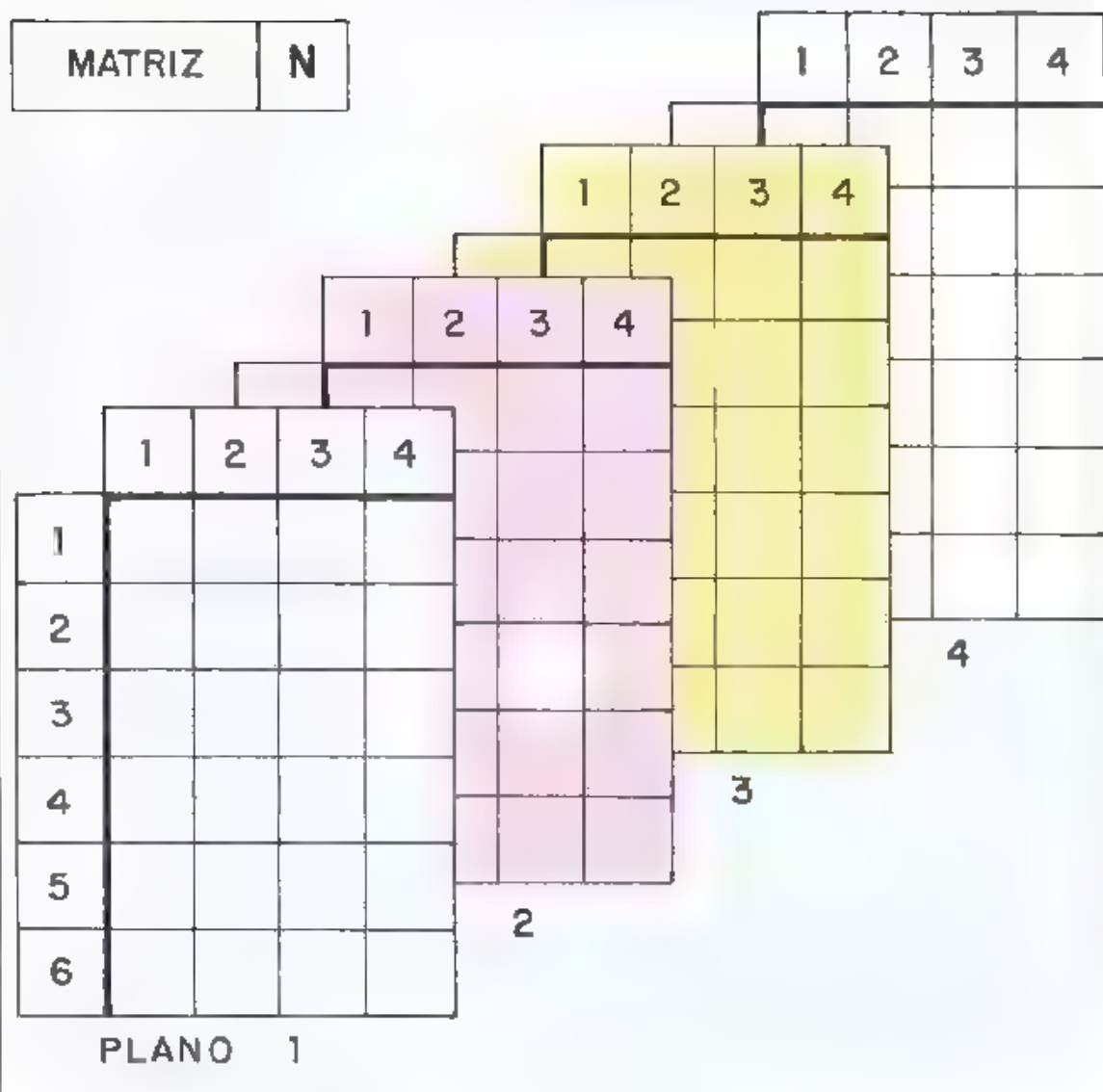
```
2 > 41
4 > 36
6 > 54
8 > 67
10 > 85
12 > 4
14 > 18
16 > 92
18 > 49
20 > 3
```

```
21 > 15
23 > 7
25 > 55
27 > 18
29 > 98
31 > 53
33 > 71
35 > 18
37 > 7
39 > 64
```

```
22 > 19
24 > 11
26 > 82
28 > 73
30 > 88
32 > 68
34 > 85
36 > 87
38 > 92
40 > 79
```

Programa 1. Generación aleatoria.

182 MICROBASIC



Matriz tridimensional N(4,6,4), elemento N(2,4,3).

Recíprocamente, puede asignarse a una variable numérica el contenido de un determinado elemento. Ejemplos.

a) Asignar a la variable «numero» el contenido del elemento 3 de la matriz «h»

```
LET numero = h(3)
```

b) Asignar a la variable «cálculo» la raíz cuadrada de elemento situado en las coordenadas 7 y 2 de la matriz «p»

El valor: 18

ha sido encontrado: 3 veces
en las posiciones:

```
> 14
> 27
> 35
```

Programa 1. Búsqueda.

LET cálculo = SQRT (7 2)

La visualización de elementos se realiza de forma similar a la asignación:

PRINT Z (7)
- PRINT J (3.5)

Cuando se desea asignar o visualizar todos los elementos de una matriz, son muy útiles los bucles «FOR...NEXT».

Ejemplo

- Dimensionar una matriz de 10 elementos, asignarles valor y posteriormente visualizarlos

```
10 REM *****
11 REM UNIDIMENSIONAL
12 REM *****
13 DIM A(10)
14 REM *****
15 REM ASIGNACION
16 REM *****
17 FOR X=1 TO 10
18   INPUT "Valor: " A(X)
19   LET A(X) = ELEMENTO
20 NEXT X
21 REM *****
```

```
*****
** VISUALIZACION **
*****
22 FOR X=1 TO 10
23   PRINT "Elemento: " A(X)
24 NEXT X
```

Para asignar valores a una matriz bidimensional, es necesario utilizar dos bucles anidados, uno que «barra» las filas y otro las columnas.

Ejemplo.

- Idem matriz «n» de 4 x 3 elementos.

```
10 REM *****
11 REM BIDIMENSIONAL
12 REM *****
13 DIM A(4,3)
14 REM *****
15 REM ASIGNACION
16 REM *****
17 FOR X=1 TO 4
18   FOR Y=1 TO 3
19     INPUT "Valor: " A(X,Y)
20     LET A(X,Y) = ELEMENTO
21   NEXT Y
22 NEXT X
23 REM *****
24 REM VISUALIZACION
25 REM *****
26 FOR X=1 TO 4
27   FOR Y=1 TO 3
28     PRINT "Elemento: " A(X,Y)
29   NEXT Y
30 NEXT X
```

Numero pares encontrados: 19

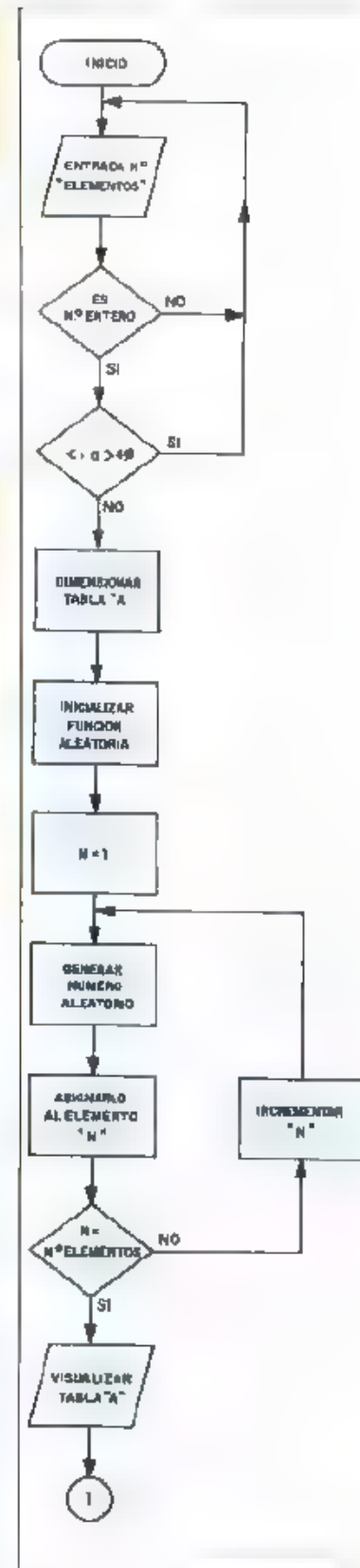
> 20	> 36
> 54	> 56
> 10	> 4
> 34	> 18
> 34	> 92
> 42	> 62
> 18	> 98
> 88	> 68
> 18	> 92
> 64	

Numero impares encontrados: 21

> 53	> 41
> 31	> 81
> 57	> 85
> 49	> 9
> 3	> 15
> 19	> 7
> 11	> 55
> 79	> 53
> 71	> 65
> 67	> 7
> 79	

Programa 1. Pares/impares.

IBM MICROBASIC



Manejo tablas
Dimensionado y asignación

Manejo de tablas

Las tablas de datos o matrices son quizás uno de los recursos más utilizados a la hora de elaborar un programa por este motivo es imprescindible que el programador conozca a fondo su manejo.

El programa n.º «1» realiza diversos tratamientos con las matrices, por este motivo va a ser explicado con bastante detalle.

Las funciones básicas que realiza el programa son

Dimensionar una matriz en función de una variable.

— Asignar a la matriz valores aleatorios

— Buscar un número determinado y visualizar cuántas veces aparece y en qué posiciones.

Calcular cuáles son los números pares e impares y visualizarlos en dos listas separadas.

— Ordenar la tabla, generada aleatoriamente, de menor a mayor

a) Dimensionado y asignación:

La línea 20 del programa permite introducir un número por teclado, éste queda asignado a la variable «elementos»; este número hace referencia a la cantidad de elementos de que va a constar nuestra tabla.

La línea 30 comprueba que el número introducido es entero, es decir, que no tiene decimales, de no ser así, el programa vuelve a pedir que introduzcamos otro número.

En la 40 se comprueba que el número está comprendido dentro del rango «1» a «40».

El dimensionado de la tabla se realiza en la línea 50. El nombre asignado es «a» y con

```

10 REM *****
   * MANEJO DE TABLAS *
   * *****
20 INPUT "Numero de elementos
>," elementos
30 IF elementos<>INT elementos
THEN GO TO 20
40 IF elementos<1 OR elementos
>40 THEN GO TO 20
50 DIM a(elementos)
60 REM *****
   * ASIGNACIÓN VALORES *
   * *****

65 RANDOMIZE
70 FOR n=1 TO elementos

80 LET contenido=INT (RND*100)
+1
90 LET a(n)=contenido
100 NEXT n
110 REM *****
   * VISUALIZACIÓN *
   * *****

120 FOR n=1 TO elementos
130 PRINT n;
132 IF n<10 THEN PRINT " ";
134 PRINT "> ",a(n),
140 NEXT n

```

una dimensión definida en la variable «elementos».

En la línea 65 se realiza la inicialización de la función aleatoria, ésta depende del tiempo que lleve conectado el ordenador (variable FRAMES).

Dentro del bucle comprendido entre las líneas 70 a 100 se realiza la asignación de valores aleatorios.

La generación del número aleatorio se realiza en la línea 80. Debido al algoritmo empleado, a función retorna valores comprendidos entre los números 1 y 100. El valor queda asignado a la variable «contenido».

En la línea 90 se realiza la asignación del valor de la variable «contenido», al elemento apuntado por la variable de control del bucle (n).

La visualización del contenido de la matriz lo lleva a cabo en el bucle formado por las líneas 120 a 140.

La línea 130 visualiza el n.º de subíndice. La 132 visualiza a continuación un espacio en blanco, si el número de subíndice es menor que 10; esto permite que la visualización de los números quede alineada.

La visualización de los contenidos de la matriz, seleccionados por la variable de con-

trof «n», lo realiza la línea 134. El formato de salida es a dos columnas

b) Búsqueda.

En la línea 160 se realiza la entrada del valor a buscar

La línea 170 comprueba si el número introducido es entero o no, y en la 180 si está comprendido dentro del rango 1 a 100.

En la 190 se inicia la variable «veces» con el valor «0». En la línea 200 se dimensiona la matriz «b», ésta va a contener las direcciones de la matriz «a» correspondientes al valor buscado

El bucle de exploración de los elementos de la matriz «a» está localizado entre las líneas 220 y 260

En la línea 230 se comprueba si el elemento direccionado por la variable de control de bucle (n, es igual al contenido de la variable «valor»

Si son iguales en la línea 240 se incrementa la variable «veces», que contiene el número de ocasiones en que el valor es encontrado

En la 250 se asigna al elemento de la matriz «b» seccionado por la variable «veces», el valor de la variable de control de bucle (n), que indica la dirección de la tabla «a» donde el valor ha sido encontrado

La línea 280 «limpia» la pantalla para poder visualizar los resultados de la búsqueda

En la 282 se comprueba si el número ha sido encontrado. En caso negativo se visualiza un mensaje indicando el hecho.

Si al menos en una ocasión el valor se encuentra, las líneas 290 a 300 se encar-

```

150 REM
      *****
      *
      * BUSQUEDA *
      *
      *****

160 INPUT "Valor a encontrar >>";valor
170 IF valor<>INT valor THEN GO TO 160
180 IF valor<1 OR valor>100 THEN GO TO 160
190 LET veces=0
200 DIM b(elementos)
210 REM
      *****
      *
      * EXPLORACION *
      *
      *****

220 FOR n=1 TO elementos
230 IF a(n)<>valor THEN GO TO 260
240 LET veces=veces+1
250 LET b(veces)=n
260 NEXT n
270 REM
      *****
      *
      * VISUALIZACION *
      *
      *****

280 CLS
282 IF veces=0 THEN PRINT "Valor ";valor;" no encontrado"; GO TO 350
290 PRINT "El valor: ",valor
300 PRINT "ha sido encontrado: ";veces;" veces"
310 PRINT "en las posiciones."

320 FOR n=1 TO veces
330 PRINT "> ",b(n)
340 NEXT n
350 PRINT #0,"Otro valor (S/N)"
360 PAUSE 0
370 LET z$=INKEY$
380 IF z$="S" OR z$="-s" THEN GO TO 150
390 IF z$="N" OR z$="-n" THEN GO TO 410
400 GO TO 360
410 CLS
412 FOR n=1 TO elementos
414 PRINT n;
416 IF n<10 THEN PRINT " ",
420 PRINT "> ",a(n),
430 NEXT n

```

gan de visualizar el «valor» y el número de «veces», y el bucle formado por las líneas 320 a 340 de visualizar las posiciones.

Las líneas 350 a 400 comprueban si se desea buscar otro valor o no.

En caso negativo se «limpia» la pantalla y se vuelve a visualizar el contenido de la matriz de origen.

c) Par-Impar

Las líneas 450 a 460 temporizan hasta que se pulsa una tecla y la 470 «limpia» la pantalla.

La 480 y 490 asignan a las variables «par» e «impar» el valor inicial 0.

La matriz «C» se dimensiona en la línea 500 y se utiliza para almacenar los números pares encontrados. La 510 dimensiona la tabla «D» que hace lo mismo con los números impares.

El bucle de búsqueda de números pares o impares está implementado entre las líneas 530 y 610.

La línea 540 asigna a la variable «división» el cociente entre el elemento de la matriz «a», direccionado por la variable de control (n), y el número dos.

La forma de calcular si el n.º es par o no se realiza en la línea 550. Un número es par cuando es divisible por dos, por lo tanto, si se multiplica la parte entera del cociente («división») por el divisor (2) y el resultado es igual al dividendo (a(n)), es que el resto es igual a cero, y por tanto, par.

Si el número es impar en las líneas 560 y 570 se incrementa la variable «impar», que almacena el número total de impares, y se asigna el número al elemento de la matriz «d», direccionada por el puntero

```

440 REM *****
      *
      * PAR/IMPAR *
      *
      *****
450 PRINT #0; 'Pulsa una tecla p
ara continuar.'
460 PAUSE 0
470 CLS
480 LET par=0
490 LET impar=0
500 DIM c(elementos)
510 DIM d(elementos)
520 REM *****
      *
      * BUSQUEDA *
      *
      *****
530 FOR n=1 TO elementos
540 LET division=a(n)/2
550 IF INT (division)*2=a(n) TH
EN GO TO 590
560 LET impar=impar+1
570 LET d(impar)=a(n)
580 GO TO 610
590 LET par=par+1
600 LET c(par)=a(n)
610 NEXT n
620 REM *****
      *
      * VISUALIZACION *
      *
      *****
625 CLS
630 PRINT "Numeros pares encont
rados. ",par
640 FOR n=1 TO par
650 PRINT "> ",c(n),
660 NEXT n
670 PRINT #0,"Pulsa una tecla p
ara continuar."
680 PAUSE 0
685 CLS
690 PRINT "Numeros impares enco
ntrados. ",impar
700 FOR n=1 TO impar
710 PRINT "> ",d(n),
720 NEXT n
730 PRINT #0,"Vuelvo a visualiz
arlos (S/N)."
740 PAUSE 0
750 LET z%=INKEY$
760 IF z%="S" OR z%="s" THEN GO
TO 620
770 IF z%="N" OR z%="n" THEN GO
TO 790
780 GO TO 740
790 CLS
800 FOR n=1 TO elementos
802 PRINT n;
804 IF n<10 THEN PRINT " ";
810 PRINT "> ",a(n),
820 NEXT n

```

«impar». En las líneas 590 y 600 se realiza la misma operación para los números pares.

La línea 625 «limpia» la pantalla. En la línea 630 se visualiza el número de pares encontrados. El bucle formado por los números 640 a 660 visualiza los valores de la matriz «C».

Las líneas 670 a 685 realizan la función de pausa y borrado.

La visualización del total de impares y sus valores, está implementado en las líneas 690 a 720.

Las líneas 730 a 780 comprueban si se desea volver a visualizar los números. En caso negativo, las líneas 790 a 820 nos vuelven a visualizar la tabla original.

d) Ordenación:

El último paso que queda es la ordenación de la tabla, para ello se ha utilizado el siguiente procedimiento.

Se compara el primer número de la tabla con el resto, si resulta que es el más pequeño, se queda donde está, pero si no, el valor más pequeño se sitúa en la primera posición y el primer número en su lugar. En el siguiente ciclo se realiza la misma operación con el número dos, este ciclo se repite tantas veces como elementos tenga la tabla menos uno, ya que el último valor queda automáticamente ordenado.

Las líneas 840 a 860 temporizan y posteriormente borran la pantalla cuando se pulsa una tecla.

El bucle formado por las líneas 870 a 980 es el dedicado a «barrer» todos los números menos uno.

La variable «menor» se inicializa en la línea 880 con el

```

830 REM
      *****
      * ORDENA *
      *****
840 PRINT #0, "Pulsa una tecla p
ara continuar"
850 PAUSE 0
860 CLS
870 FOR x=1 TO elementos-1
880 LET menor=x
890 FOR y=x+1 TO elementos
900 IF a(menor)>a(y) THEN LET m
enor=y
910 NEXT y
920 LET cambio=a(x)
930 LET a(x)=a(menor)
940 LET a(menor)=cambio
950 PRINT x,
960 IF x<10 THEN PRINT " ",
970 PRINT "> ", a(x),
980 NEXT x
990 PRINT elementos,
1000 IF elementos<10 THEN PRINT
" ",
1010 PRINT "> ", a(elementos)
1020 PAUSE 0

```

valor correspondiente de la variable de control del bucle (x).

El bucle de comparación se encuentra implementado en las líneas 890 a 910.

Al final del bucle la variable «menor» contiene el menor número de los comparados.

En las líneas 920 a 940 se realiza el intercambio entre los elementos direccionados

por los subíndices «x» y «menor»; este intercambio se apoya en la variable «cambio» para poder realizarlo.

La visualización de la posición y del valor ordenado lo ejecutan las líneas 950 a 970.

Para finalizar la visualización del último elemento (mayor), lo llevan a cabo las líneas 990 a 1010.

1	>	3
3	>	9
7	>	11
9	>	18
11	>	18
13	>	20
15	>	34
17	>	36
19	>	42
21	>	53
23	>	54
25	>	56
27	>	64
29	>	68
31	>	70
33	>	81
35	>	87
37	>	88
39	>	92

2	>	4
4	>	7
6	>	10
8	>	15
10	>	18
12	>	19
14	>	31
16	>	34
18	>	41
20	>	49
22	>	53
24	>	55
26	>	62
28	>	67
30	>	71
32	>	79
34	>	85
36	>	87
38	>	90
40	>	96

Programa 1. Ordenación.

PROGRAMA 2

```

10 REM *****
* CURSO BASIC *
* *****
* CARTAS *
* *****

15 BORDER 1: PAPER 4 INK 0 C
L3
20 PRINT AT 3,0, " (C) MICRO
HOBBY SEMANAL
22 PRINT AT 8,13, "JUEGO"
24 PRINT AT 12,7, "LAS SIETE Y
MEDIA"
26 PRINT H, 18,0, " POR KATA
EL Prades
28 PAUSE 150
30 REM

*****
* CREDITO/APUESTA *
*****

40 LET credito=10000
50 LET cred ordenador credito
60 LET cred jugador credito
70 LET apuesta=100
75 LET manos=1
80 REM

*****
* DIMENSIONADO *
*****

90 DIM m(40,2)
100 DIM b(10,4)
110 DIM c(40,2)
115 DIM v(10)
120 DIM t$(10,4,20)
125 DIM r$(1,5)
130 REM

*****
* ASIGNACION *
*****

140 FOR p=1 TO 4
142 RESTORE 250+p
144 READ a$
150 RESTORE 260
152 FOR n=1 TO 10
154 READ b$
156 LET t$(n,p)=b$+" de "+a$
160 NEXT n
170 NEXT p
250 REM

*****
* DATOS DE CARTAS *
*****

251 DATA "OROS"
252 DATA "CORAS"
253 DATA "ESPACAS"
254 DATA "BASTOS"
255 DATA "AS"
256 DATA "DOS", "TRES", "CUATRO", "CINCO", "SEIS", "SIETE", "SOTA", "CABALLO", "REY"
300 REM

*****
* VALOR *
*****

310 FOR x=1 TO 7
320 LET v(x)=x
330 NEXT x
340 FOR a=0 TO 10
350 LET v(x)=0.5
360 NEXT x

```

```

365 GO SUB 2100
370 REM

*****
* BARAJEAR *
*****

372 RANDOMIZE
380 CLS
390 PRINT AT 3,0, " ESPERA "
400 PRINT AT 1,0, " VO H BA
RAJEAR
410 FOR i=1 TO 40
420 GO SUB 490
430 IF n numero palo=1 THEN GO
TO 420
440 LET m(x,1)=numero
450 LET m(x,2)=palo
460 LET b(numero,palo)=1
470 NEXT i

480 GO TO 570
490 REM

*****
* CARTA ALEATORIA *
*****

500 LET numero=INT (RND*10)+1
510 LET palo=INT (RND*4)+1
520 RETURN
530 REM

*****
* CORTAR EL MAZO *
*****

540 CLS
550 PRINT AT 3,0, " YA ES
H"
560 PRINT AT 20,1, "Por que n me
ro a, seras cortar?
570 INPUT "LINE n$
572 IF n$="" THEN GO TO 570
574 FOR i=1 TO LEN n$
576 IF n$(i)="" OR n$(i)="" THEN GO TO 570
578 NEXT i
580 LET numero=VAL n$
590 IF numero<1 OR numero>40 TH
EN GO TO 570
592 CLS
594 PRINT AT 3,0, " ESPERA "
600 LET resto=40-numero
610 FOR x=1 TO numero
620 LET c(x+resto,1)=m(x,1)
630 LET c(x+resto,2)=m(x,2)
640 NEXT x
650 FOR x=numero+1 TO 40
660 LET c(x-numero,1)=m(x,1)
670 LET c(x-numero,2)=m(x,2)
675 NEXT x
680 PRINT AT 3,0, " PODEMOS E
PEZAR A JUGAR
690 PRINT AT 18,0, " !! BUEN
A SIERTE "
700 PRINT #0, " Pulsa una tecla
para continuar"
710 PAUSE 0
715 CLS
720 REM

*****
* INICIAL JUGADOR *
*****

722 PRINT AT 0,0, " TO JUEGAS

```

```

730 LET puntos_jugador=0
740 LET carta=1
750 PRINT AT 3,0,
760 GO SUB 800
770 LET puntos_jugador=valor ca
780 LET carta=2
790 GO TO 850
800 REM
*****
* PRESENTACION *
*****
810 LET numero=(carta,1)
820 LET palo=carta,2)
830 PRINT "numero,palo)
840 LET valor carta=valor numero
850 RETURN
860 FC

```

```

*****
* MAS JUGADOR *
*****

```

```

870 PRINT #0, AT 1,3, "Quieres q
880 IF INKEY$=" " THEN GO TO 880
890 LET z$=INKEY$
900 IF z$="5" OR z$="s" THEN GO
910 IF z$="y" OR z$="n" THEN LC
920 GO TO 880
930 INPUT 0
940 FOR x=1 TO 25 NEXT x
950 LET carta=carta+1
960 LET puntos_jugador=puntos_j
970 IF puntos_jugador>7.5 THEN
980 IF puntos_jugador=7.5 THEN
990 GO TO 1070
1000 PRINT AT 0,0, "LO SIEMPRE,
1010 LET p$="STR$ puntos_jugad
1020 PRINT AT 21,0, INVERSE 1, "
1030 LET ganador=0
1040 PRINT #0, "Pulsa una tecla
1050 PAUSE 0
1060 GO TO 1630
1070 REM

```

```

*****
* 7 1/2 JUGADOR *
*****

```

```

1080 PRINT AT 0,0, "ARriba SIET
1090 PRINT AT 21,0, "ESPERA A Q
1100 LET indice carta
1110 PRINT #0, "Pulsa una tecla
1120 PAUSE 0
1130 GO TO 1200
1140 REM

```

```

*****
* TANTOS JUGADOR *
*****

```

```

1150 LET p$(1)=STR$ puntos_jugad
1160 PRINT AT 0,0, INVERSE 1, "
1170 LET p$(1)=STR$ puntos_jugad
1180 PRINT AT 21,0, "ARriba SIET
1190 PRINT AT 21,0, "ESPERA A Q

```

```

1175 INPUT 0
1180 PRINT #0, "Pulsa una tecla
1190 PAUSE 0
1200 REM

```

```

*****
* INICIAL ORDENADOR *
*****

```

```

1210 CLS
1220 LET puntos_ordenador=0
1230 LET carta=2
1240 PRINT AT 0,0, "LO JUEGO

```

```

1250 PRINT AT 3,0, "
1260 LET y=5
1270 GO SUB 800
1280 LET puntos_ordenador=valor
1290 LET carta=indice
1300 REM

```

```

*****
* MAS JUGADOR *
*****

```

```

1300 IF puntos_ordenador>7.5 THE
1310 IF puntos_ordenador=7.5 THE
1320 IF puntos_ordenador>7.5 THE
1330 LET ganador=1
1340 PRINT AT 0,0, "ME HE PASAD
1350 LET p$(1)=STR$ puntos_orden
1360 PRINT AT 21,0, INVERSE 1, "
1370 LET p$(1)=STR$ puntos_orden
1380 PRINT #0, "Pulsa una tecla
1390 PAUSE 0
1400 GO TO 1630
1410 REM

```

```

*****
* 7 1/2 ORDENADOR *
*****

```

```

1410 LET ganador=0
1420 PRINT AT 0,0, "ARriba SIET
1430 PRINT #0, "Pulsa una tecla
1440 PAUSE 0
1450 GO TO 1630
1460 REM

```

```

*****
* CONTINUA *
*****

```

```

1460 IF puntos_jugador=7.5 THEN
1470 IF puntos_ordenador=6 THEN
1480 GO TO 1490
1490 PRINT AT 21,0, "FIDELITY
1500 PAUSE 30
1510 PRINT AT 21,0,

```

```

1500 FOR x=1 TO 50 NEXT x
1510 LET carta=carta+1
1520 PRINT AT 4,0, "
1530 GO SUB 800
1540 LET puntos_ordenador = punt

```

```

os ordenador+valor carta
1550 GO TO 1300
1560 REM

*****
* TANTOS ORDENADOR *
*****

1570 PRINT AT 0,0, "ME QUEDO"
1580 LET p$(1)=STR$ puntos orden
ador
1590 PRINT AT 21,0, INVERSE 1
HE HECHO UN TOTAL DE p$ 1'
1590 PRINT #0, "Pulsa una tecla
para continuar"
1600 PAUSE 0
1600 REM

```

```

*****
* COMPARACION *
*****

```

```

1610 IF puntos ordenador =puntos
jugador THEN LET ganador=0
1620 IF puntos ordenador<puntos
jugador THEN LET ganador=1
1630 REM

```

```

*****
* GANADOR *
*****

```

```

1635 CLS
1640 IF ganador=0 THEN PRINT AT
3,0, "HE GANADO!!"
1640 PRINT AT 13,0, "LO SIENTO, O
TRA VEZ SEAA"
1650 IF ganador=1 THEN PRINT AT
3,0, "HAS GANADO"
1650 PRINT AT 13,0, "LA PROXIMA G
ANARE YO"
1660 PRINT AT 8,4, "Puntos jugado
r....., puntos jugador
1670 PRINT AT 13,4 "Puntos Orden
ador....."
1680 IF puntos ordenador=0 THEN
PRINT GO TO 1700
1690 PRINT puntos ordenador
1700 PRINT #0, "Pulsa una tecla
para continuar"
1710 PAUSE 0
1720 REM

```

```

*****
* CALCULOS *
*****

```

```

1730 IF ganador=0 THEN GO TO 170
0
1740 IF puntos jugador>1.5 THEN
LET volte=go to 1700
1750 LET doble=1
1760 LET cred ordenador=cred ord
enador+apuesta*dob
1770 LET cred jugador=cred jugad
or+apuesta*doble
1780 GO TO 1810
1790 LET cred ordenador=cred ord
enador+apuesta
1800 LET cred jugador=cred jugad
or+apuesta
1810 REM

```

```

*****
* RESULTADOS *
*****

```

```

1815 CLS
1820 PRINT AT 7,3, "Manos jugadas
Manos"
1830 PRINT AT 12,3 "Credito Jug
ador....., cred jugador
1840 PRINT AT 17,3, "Credito Orde
nador....., cred ordenador
1850 IF cred ordenador=0 THEN PR
INT AT 3,0, "HAS ROTO LA BANCA
PAUSE 0 STOP
1860 IF cred jugador=0 THEN PRIN
T AT 3,0, "SE ACABO T. DINERO
PAUSE 0 STOP
1870 PRINT AT 3,0, "CREDITOS"
1875 REM

```

```

*****
* CONTINUAR? *
*****

```

```

1880 PRINT #0, "Quieres jugar ot
ra mano (S/N)"
1890 PAUSE 0
1900 LET z$=INKEY$
1910 IF z$="S" OR z$="s" THEN GO
TO 1940
1920 IF z$="N" OR z$="n" THEN IN
PUT 0 PAUSE 0 STOP
1930 GO TO 1890
1940 INPUT 0
1945 LET manos=manos+1
1950 FOR a=1 TO 10
1960 FOR b=1 TO 4
1970 LET b(a,b)=0

```

```

1980 NEXT b
1990 NEXT a
2000 GO TO 380
2100 REM

```

```

*****
* INSTRUCCIONES *
*****

```

```

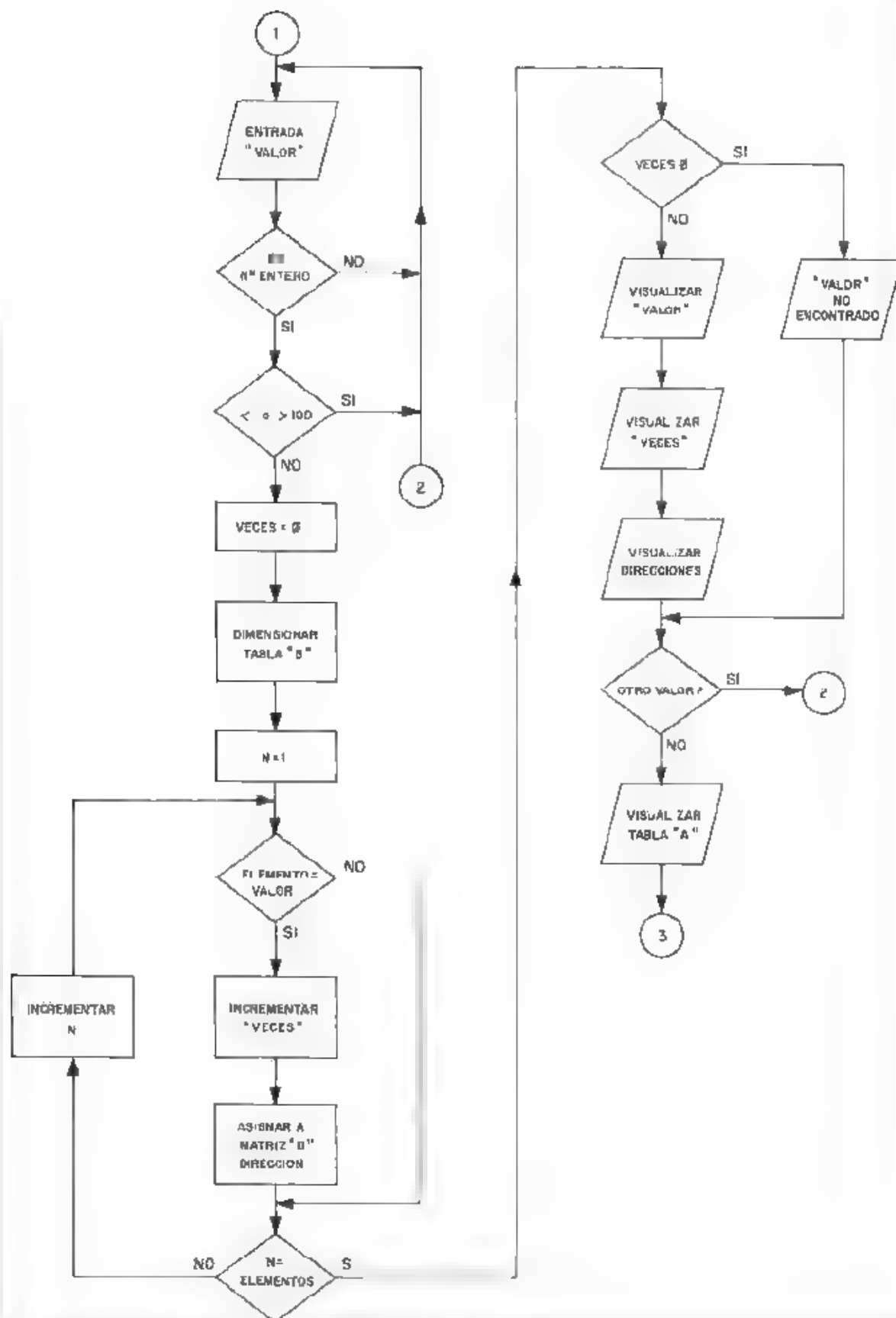
2110 PRINT #0, AT 1,6, "Instruccio
nes"
2120 PAUSE 0
2130 LET z$=INKEY$
2140 IF z$="S" OR z$="s" THEN GO
TO 2150
2150 IF z$="N" OR z$="n" THEN RE
TURN
2160 GO TO 2120
2165 CLS
2170 PRINT AT 0,0, "INSTR
UCCIONES"

```

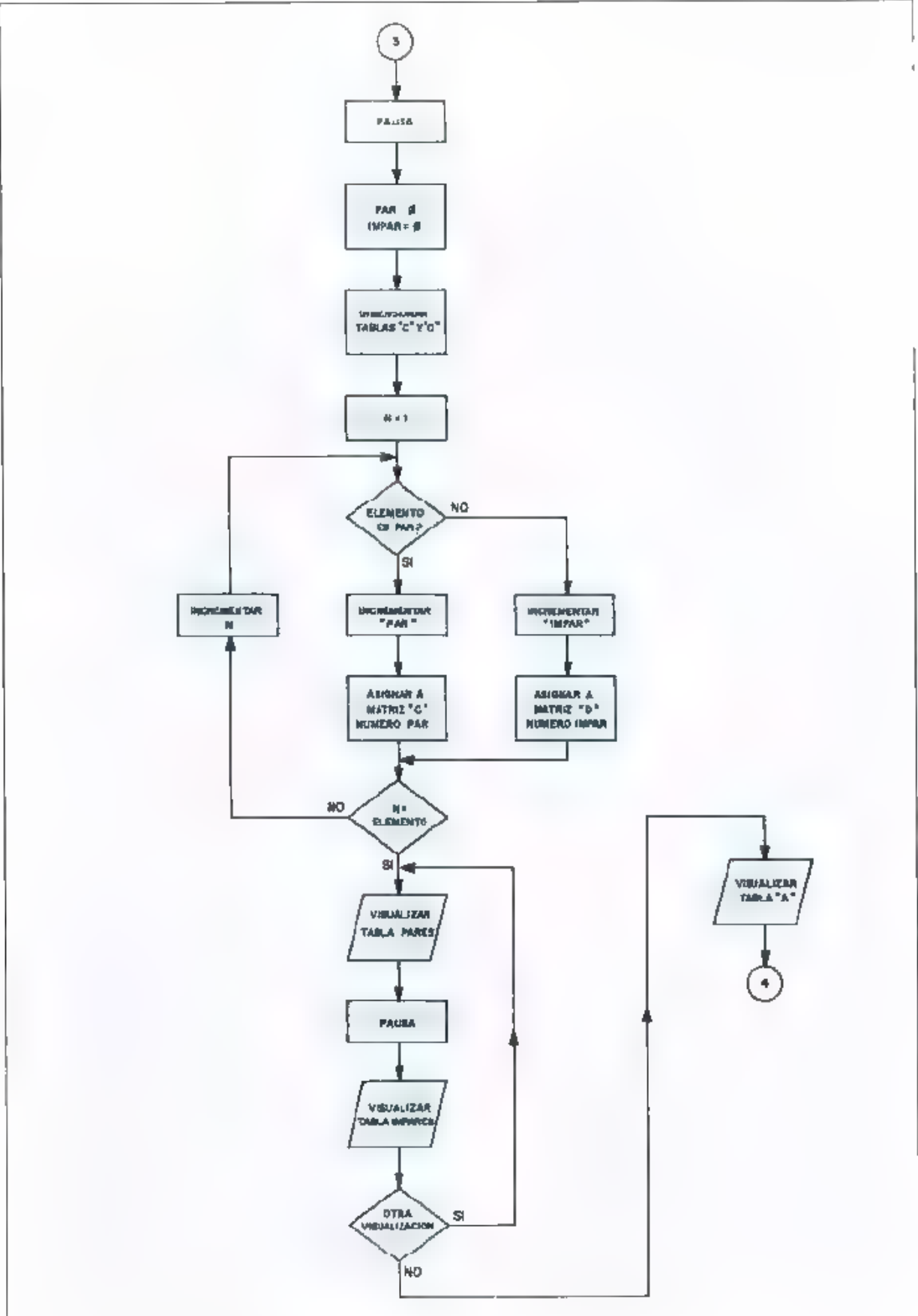
```

2180 PRINT "Este programa es t
a basado en el conocido j
e
o de naipes Las
2190 PRINT AT 10,1, "Tienes un cr
edito inicial de 10.0
000 pts
2200 PRINT AT 15,1, "Las apuestas
son de 100 pts.
2210 PRINT AT 18,1, "Si tienes 7
1/2 y ganas obtiene un benefi
cio de 200 pts.
2220 PRINT #0, "Pulsa una tecla
para continuar"
2230 PAUSE 0
2240 RETURN

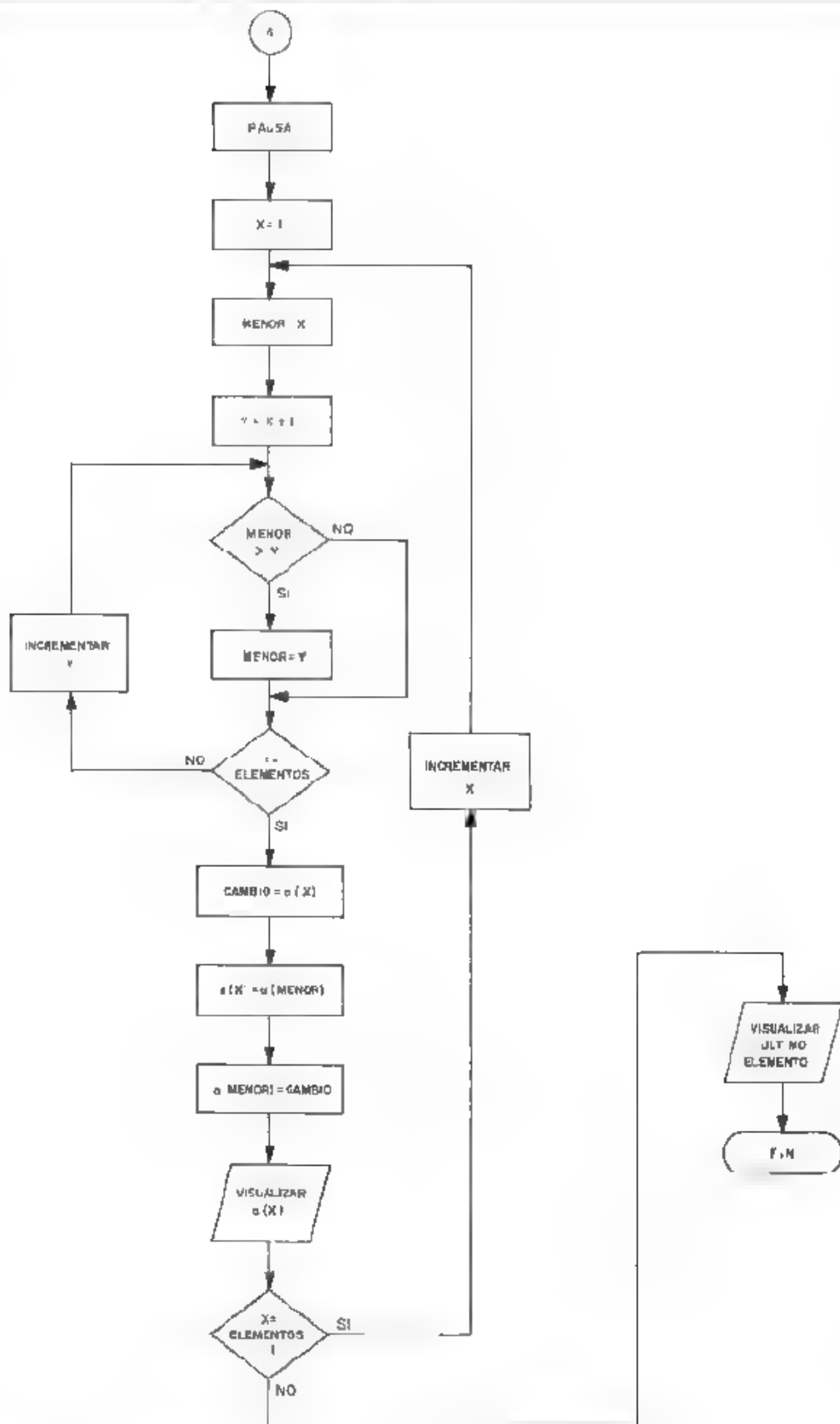
```



Manejo Tablas. Búsqueda.



Manejo tablas. PAR/IMPAR.



Manejo tablas. Ordenación.

Matrices de cadena

La estructura del dimensionado de matrices de cadena es el siguiente.

SENTENCIA	ARGUMENTO
DIM	letra \$ (il sis de valores)

Ejemplos

- DIM a\$ (30)
- DIM x\$ (2, 20)
- DIM n\$ (3, 7, 5)
- DIM z\$ (8, 4, 5, 10)

El nombre de una matriz de cadena está formado por una sola letra seguida del símbolo «dolar» (\$). A diferencia de las numéricas, no pueden compartir el mismo nombre una matriz de cadena y una variable del mismo tipo.

Al dimensionar una matriz de cadena, el contenido de sus elementos queda inicializado con espacios en blanco.

Ejemplo:

```
10 DIM a$ (10, 7)
20 FOR i = 1 TO 10
30 PRINT INVERSE 1 a$ (i)
40 NEXT i
```

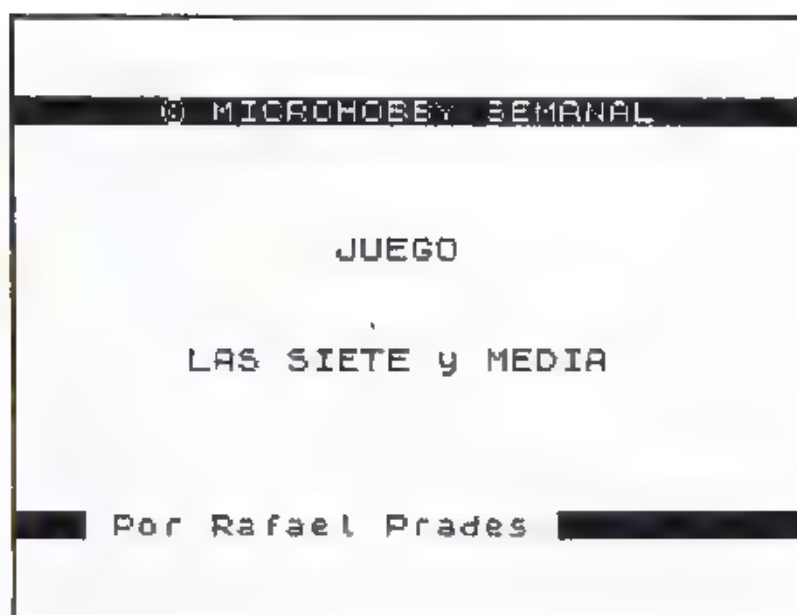
El argumento «INVERSE 1» permite visualizar los espacios, ya que intercambia el color de «tinta» por el de «papel».

Existen ciertas diferencias entre el dimensionado de matrices numéricas y el de cadenas, debido a que es preciso indicar de cuantos caracteres va a constar cada elemento.

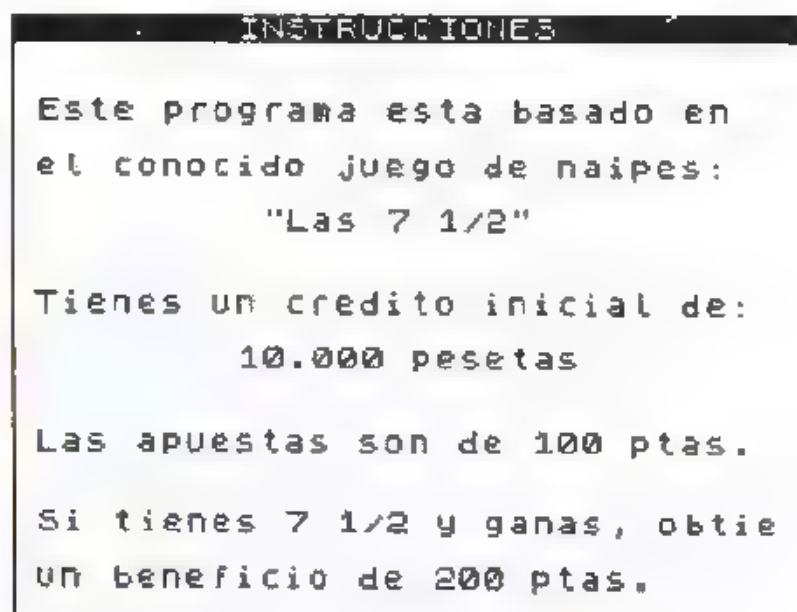
Ejemplos:

a) Dimensionar una matriz de un elemento de 20 caracteres.

```
DIM n$ (20)
```



Programa 2. Carátula.



Programa 2. Ilustraciones.

Las características de esta matriz son similares a las de una variable de cadena.

b) Matriz «unidimensional» de 100 por 20 elementos, con 7 caracteres cada uno

```
DIM Z$ (7, 15)
```

c) Matriz «bidimensional»

de 100 por 20 elementos, con 7 caracteres cada uno

```
DIM J$ (100, 20, 7)
```

d) Matriz «tridimensional» formada por tres planos de 10 por 4; el máximo de caracteres por elemento es cinco.

```
DIM T$ (3, 10, 4, 5)
```

```

LO SIENTO, TE HAS PASADO

> AS de BASTOS
> SOTA de ESPADAS
> TRES de COPAS
> CABALLO de BASTOS
> CUATRO de BASTOS

HAS HECHO UN TOTAL DE 3

```

Programa 2. Juegos.

Asignación

En la asignación de valores a los elementos de una matriz de cadena, no es necesario hacer referencia al último subíndice dimensionado es decir, el que indica el número de caracteres.

Ejemplo:

```

10 DIM W$(10, 5)
20 LET W$(3) = "PEPE"
30 PRINT W$(3)

```

Cuando se indica dicho subíndice, es que se hace referencia a un determinado carácter del elemento seleccionado. Siguiendo con el mismo ejemplo:

```

40 LET W$(3, 4) = "A"
50 PRINT W$(3, 4)
60 PRINT W$(3)

```

La asignación de la línea 40 quiere expresar: Asignar al carácter cuatro del elemento tres, perteneciente a la matriz «W\$», el valor de cadena «A»; los demás caracteres

quedan con el mismo valor.

La asignación de valores es del tipo *procusteano*, es decir, que cuando la longitud de la cadena es menor que el número de caracteres reservados, el elemento de la matriz se rellena con espacios y por el contrario, cuando es más larga se recorta. En la página 45 (Asignación de subcadenas) viene explicado este tipo de asignación.

Ejemplos:

a) Rellenado con espacios.

```

10 REM *****
11 RELENADO
12 *****

20 DIM M$(12, 10)
30 FOR I=1 TO 12
40 READ
50 LET M$(I) =
60 PRINT INVERSE I, M$, INVERSE
70 PRINT INVERSE I, M$
80 NEXT I
90 REM *****
100 DATA
110 DATOS
120 *****

```

```

100 DATA "ENERO", "FEBRERO", "MAR",
200 "ABRIL", "MAYO", "JUNIO", "JULI",
300 "AGOSTO", "SEPTIEMBRE", "OCTUBA",
400 "NOVIEMBRE", "DICIEMBRE"

```

Los elementos de la matriz «M\$» están dimensionados de 10 caracteres, todos los

datos (meses) que tengan *menor* longitud quedan rellenos con espacios, de esta manera se mantiene la longitud total (10).

b) Recortado

```

10 REM *****
11 RECORCADO
12 *****

20 DIM A$(10, 5)
30 FOR I=1 TO 7
40 READ
50 LET A$(I) =
60 PRINT INVERSE I, A$, INVERSE
70 PRINT INVERSE I, A$
80 NEXT I
90 REM *****
100 DATA
110 DATOS
120 *****
130 DATA "LUNES", "MARTES", "MIÉ",
140 "COLES", "JUEVES", "VIERNES", "SÁB",
150 "DOMINGO"

```

En este otro ejemplo, los elementos de la matriz «A\$» tienen una longitud fija de cinco caracteres, cualquier dato (días) que tenga una *mayor* longitud queda recortado a este valor.

Fragmentación

De la misma manera que las variables de cadena, las matrices pueden fragmentarse (VER pag. 43).

Asignemos primero un valor a un elemento de la matriz «A\$», y veamos posteriormente unos ejemplos.

```

10 DIM A$(10, 10)
20 LET A$(5) = "ORDENADOR"
30 PRINT A$(5)

```

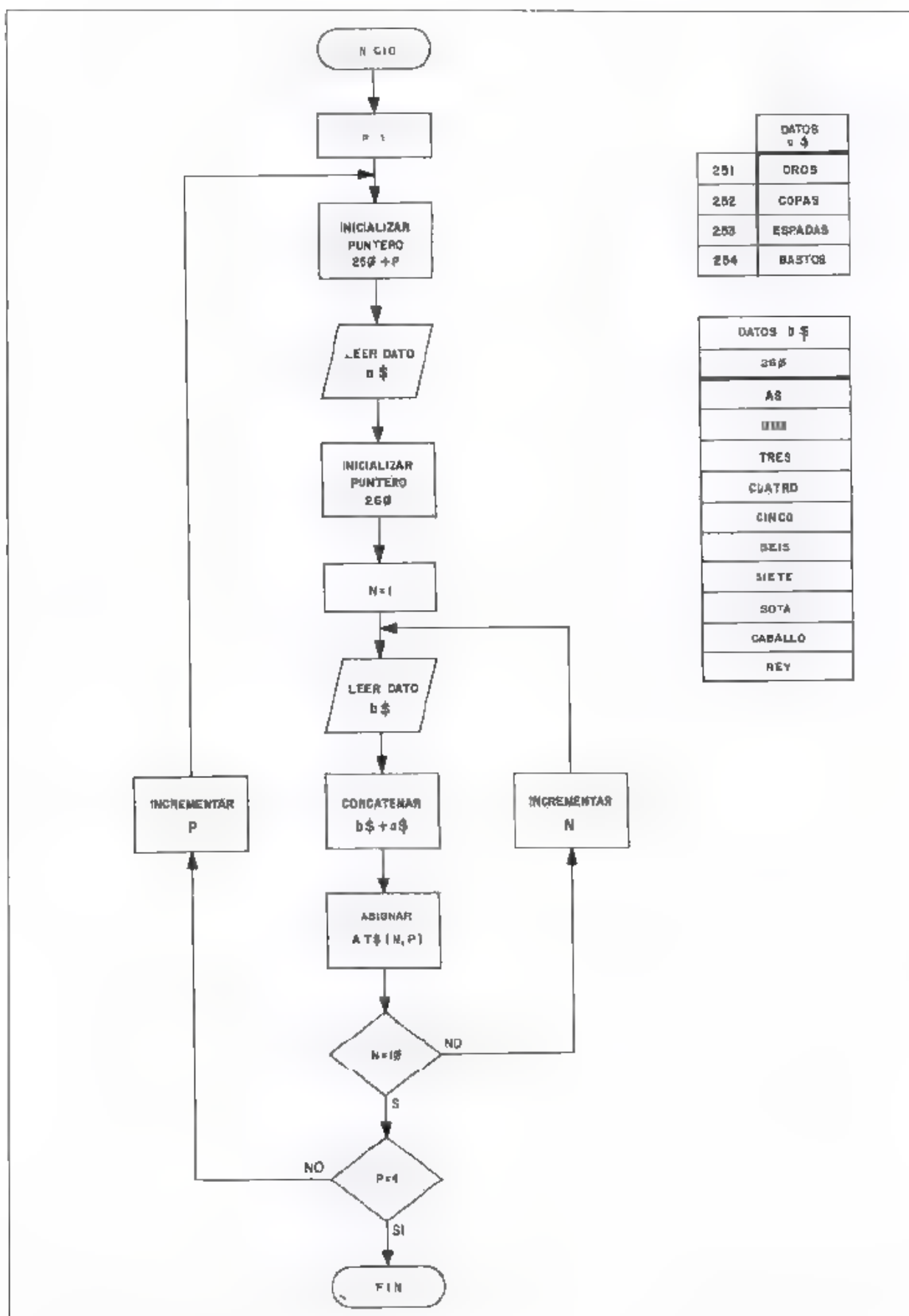
a) Asignar a los caracteres 3 a 5 el valor «POS»

```

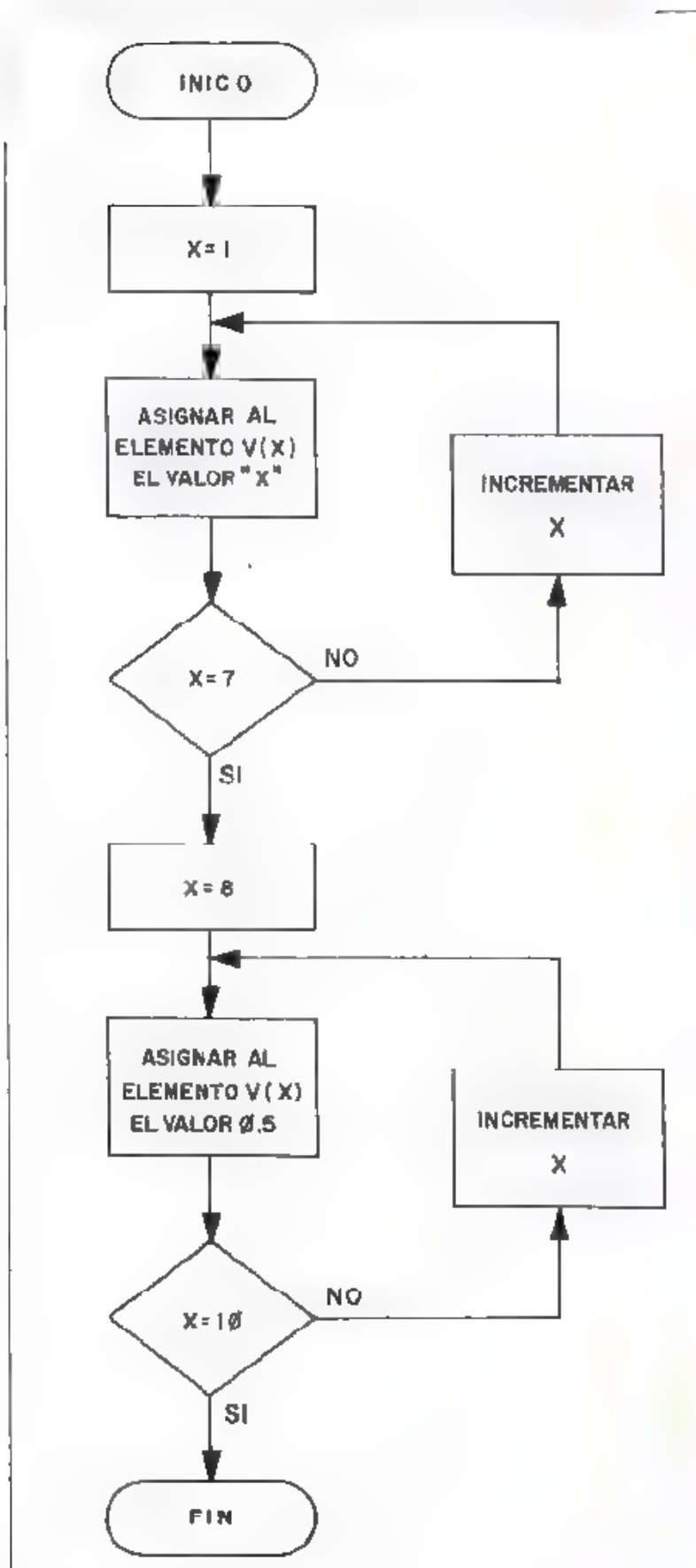
40 LET A$(5, 3 TO 5) = "POS"
50 PRINT A$(5)

```

b) Asignar a los cuatro primeros caracteres la cadena «TRIN».



Rutina asignación de naipes.



Rutina asignación de naipes.

```

60 LET A$ 15 TO 4 = "TRIN"
70 PRINT A$ 5;

```

c) Asgnar a partir del quinto carácter la cadena "CHAR".

```

80 LET A$ 5 5 TO) = "CHAR"
90 PRINT A$ 5;

```

d) Asgnar al carácter número 9 el valor "E".

```

100 LET A$ 9 9) = "E"
10 PRINT A$ 15

```

La fragmentación permite utilizar tablas bidimensionales con diversos campos. Supongamos que deseamos crear una tabla para almacenar los datos referentes a nuestros programas, ésta podría tener cuatro campos:

```

PROGRAMA
CIN A
CARA
TIPO

```

En el primer campo almacenaríamos el nombre del programa; en el segundo, el nombre de la cinta donde se encuentra; en el tercero en qué cara de la cinta, y en el último, el tipo de programa, es decir, si es un juego, utilidad, etc.

La cantidad de caracteres de cada campo va a ser la siguiente:

CAMPO	CARACT
PROGRAMA	10
CINTA	10
CARA	1
TIPO	

El campo «CARA» es de un solo carácter ya que va a con-

La forma de dimensionar la matriz, suponiendo que el número máximo de programas es 100, sería:

¿Cómo con un dimensionado de este tipo se puede acceder a cada campo? En total, la cantidad de información que necesitamos por programa, es de 22 caracteres ($10 + 10 + 1 + 1$). La forma de acceder a cada campo, por ejemplo del elemento 1, sería la siguiente

```
PRINT "Programa " p$ (1 TO 10)
PRINT "Coto " p$ (11 TO 20)
PRINT "Cala " p$ (21)
PRINT "Tipo " p$ (22)
```

ENERO	
FEBBRE	RA
MARZO	
ABRIL	
MAYO	
JUNIO	
JULIO	
AGOSTO	
SEPTIEMBRE	
OCTUBRE	
NOVIEMBRE	
DICIEMBRE	

1. **PLANNING**
 2. **CONCEPTS**
 3. **DESIGN**
 4. **CONSTRUCTION**
 5. **OPERATION**
 6. **MAINTENANCE**
 7. **REPAIR**
 8. **REPLACEMENT**
 9. **RENOVATION**
 10. **RECONSTRUCTION**
 11. **REDEMPTION**
 12. **REDEMPTION**
 13. **REDEMPTION**
 14. **REDEMPTION**
 15. **REDEMPTION**
 16. **REDEMPTION**
 17. **REDEMPTION**
 18. **REDEMPTION**
 19. **REDEMPTION**
 20. **REDEMPTION**
 21. **REDEMPTION**
 22. **REDEMPTION**
 23. **REDEMPTION**
 24. **REDEMPTION**
 25. **REDEMPTION**
 26. **REDEMPTION**
 27. **REDEMPTION**
 28. **REDEMPTION**
 29. **REDEMPTION**
 30. **REDEMPTION**
 31. **REDEMPTION**
 32. **REDEMPTION**
 33. **REDEMPTION**
 34. **REDEMPTION**
 35. **REDEMPTION**
 36. **REDEMPTION**
 37. **REDEMPTION**
 38. **REDEMPTION**
 39. **REDEMPTION**
 40. **REDEMPTION**
 41. **REDEMPTION**
 42. **REDEMPTION**
 43. **REDEMPTION**
 44. **REDEMPTION**
 45. **REDEMPTION**
 46. **REDEMPTION**
 47. **REDEMPTION**
 48. **REDEMPTION**
 49. **REDEMPTION**
 50. **REDEMPTION**
 51. **REDEMPTION**
 52. **REDEMPTION**
 53. **REDEMPTION**
 54. **REDEMPTION**
 55. **REDEMPTION**
 56. **REDEMPTION**
 57. **REDEMPTION**
 58. **REDEMPTION**
 59. **REDEMPTION**
 60. **REDEMPTION**
 61. **REDEMPTION**
 62. **REDEMPTION**
 63. **REDEMPTION**
 64. **REDEMPTION**
 65. **REDEMPTION**
 66. **REDEMPTION**
 67. **REDEMPTION**
 68. **REDEMPTION**
 69. **REDEMPTION**
 70. **REDEMPTION**
 71. **REDEMPTION**
 72. **REDEMPTION**
 73. **REDEMPTION**
 74. **REDEMPTION**
 75. **REDEMPTION**
 76. **REDEMPTION**
 77. **REDEMPTION**
 78. **REDEMPTION**
 79. **REDEMPTION**
 80. **REDEMPTION**
 81. **REDEMPTION**
 82. **REDEMPTION**
 83. **REDEMPTION**
 84. **REDEMPTION**
 85. **REDEMPTION**
 86. **REDEMPTION**
 87. **REDEMPTION**
 88. **REDEMPTION**
 89. **REDEMPTION**
 90. **REDEMPTION**
 91. **REDEMPTION**
 92. **REDEMPTION**
 93. **REDEMPTION**
 94. **REDEMPTION**
 95. **REDEMPTION**
 96. **REDEMPTION**
 97. **REDEMPTION**
 98. **REDEMPTION**
 99. **REDEMPTION**
 100. **REDEMPTION**

ENERO
FEBBERO
MARZO
APRIL
MAYO
JUNIO
JULIO
AGOSTO
SEPTIEMBRE
OCTUBRE
NOVIEMBRE
DICIEMBRE

[illegible]**MICROBASIC 199**

2 Variable not found

b) Cuando uno de los subíndices es negativo o mayor que «65535» aparece

8 Integer out of range

Ejemplo:

— PRINT a\$(12, 1)

— PRINT a\$(67000,3)

c) Cuando un subíndice está fuera de los límites del dimensionado de una matriz, se visualiza

3 Subscript wrong

Por ejemplo, en la matriz «b\$(20, 10)»

LET b\$(0,5)

— PRINT b\$(7,11)

d) Cuando se dimensiona una tabla demasiado grande que ocupa toda la memoria destinada a los programas «BASIC», aparece el error

4 Out of memory

Grabación de datos

Con el comando «SAVE» es posible almacenar o grabar en cinta las tablas generadas en alguno de nuestros programas; también pueden ser cargadas, posteriormente en el ordenador, con el comando «LOAD».

La sintaxis es la siguiente:

SAVE "nombre" DATA letra ()

en el caso de matrices numéricas, y

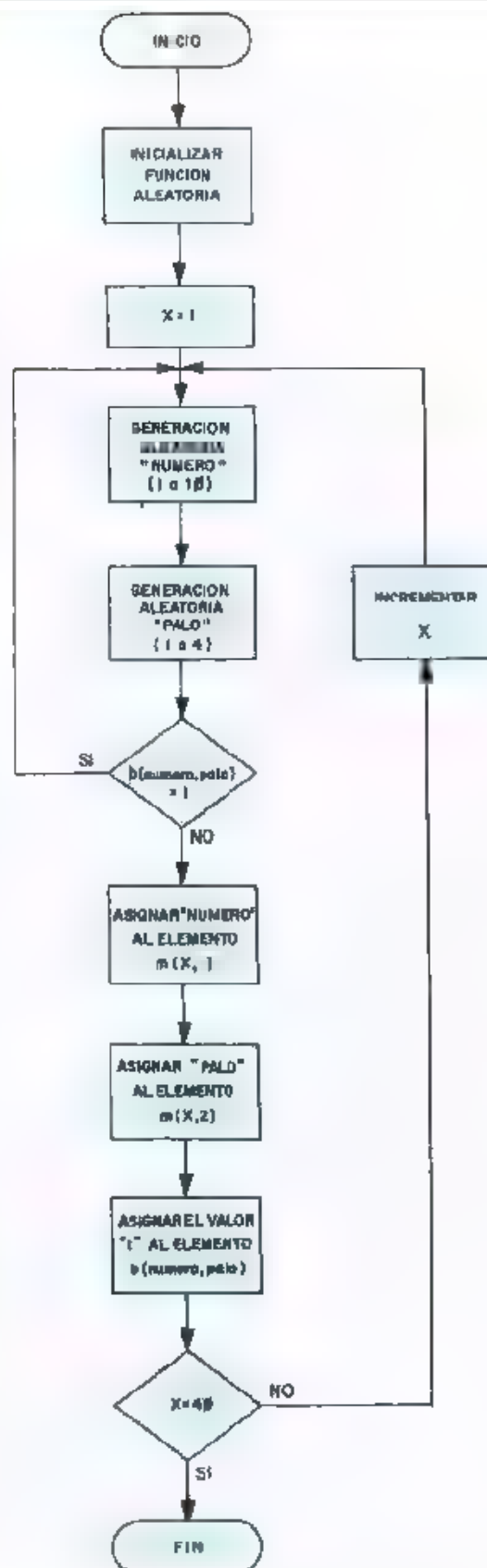
SAVE "nombre" DATA letra \$()

en el de cadena.

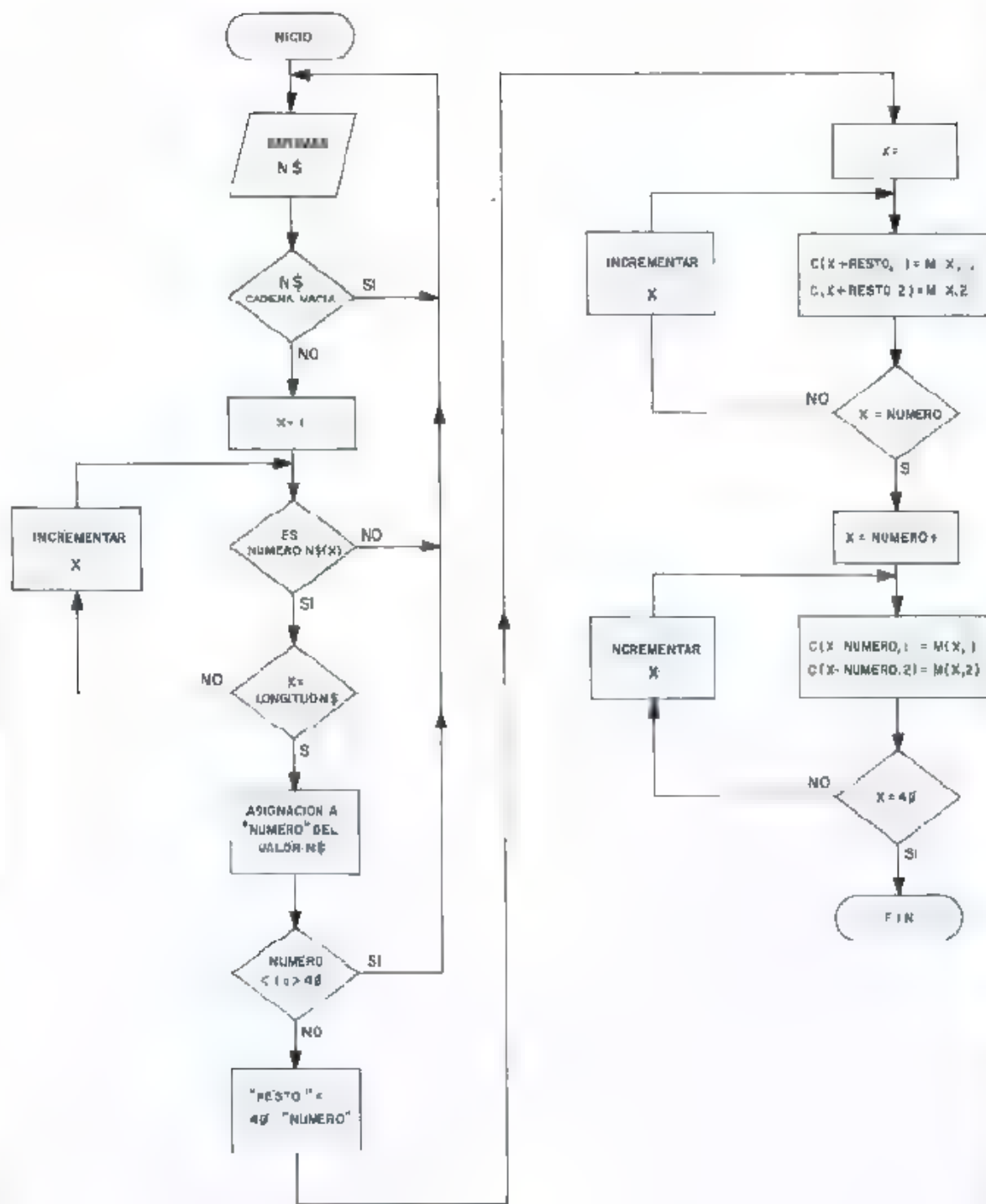
El *nombre*, entre comillas, es el que se asigna a la tabla de datos en la cinta, y la *letra* corresponde al asignado al dimensionar la matriz.

Si desea verificar o cargar una tabla, previamente salva-da, sustituya el comando «SAVE» por «VERIFY» o «LOAD».

MICROBASIC 200



Rutina «barajar» los naipes.



Rutina "cortar" el mazo.

Programa

El programa numero «2» permite jugar con el ordenador, al conocido juego de cartas «Las Siete y Media»

El ordenador hace el papel de «banca» y por tanto, baraja y reparte las cartas; nosotros tenemos la posibilidad de indicar por qué número deseamos «cortar» la baraja

Después de repartir las dos primeras cartas, una para él y otra para nosotros, nos irá preguntando sucesivamente si deseamos otra carta o no.

Al final de la partida se presentan los tantos obtenidos por cada uno de los dos jugadores y si deseamos jugar otra «mano».

Para dar mayor emoción a la partida, al principio de ésta, tenemos un crédito de «10 000» ptas. Las apuestas son de «100» ptas., ganamos el doble si obtenemos «siete y media» y el ordenador no, ya que en igualdad de puntos gana la banca

BUENA SUERTE!

Hay una serie de matrices utilizadas en la confección del programa, que se encuentran dimensionadas en las líneas 90 a 125.

La matriz «T\$» almacena el nombre de cada carta (as deoros, dos de ..., etc.) Es una tabla bidimensional de «10» por «4», ya que la baraja consta de 4 «palos» de diez cartas. La tercera dimensión (20) indica el número máximo de caracteres de cada elemento, y se encuentra ligeramente sobredimensionada ya que con dieciocho hubiera bastado, por que el contenido de mayor longitud va a ser «cabo lo de espadas» (16 letras + 2 espacios).

10	BLOQUE 1
20	
30	
40 STOP	
50	BLOQUE 2
60	
70	
80 STOP	
90	BLOQUE 3
100	
110 STOP	
120	
130	BLOQUE 4
140	
150 STOP	

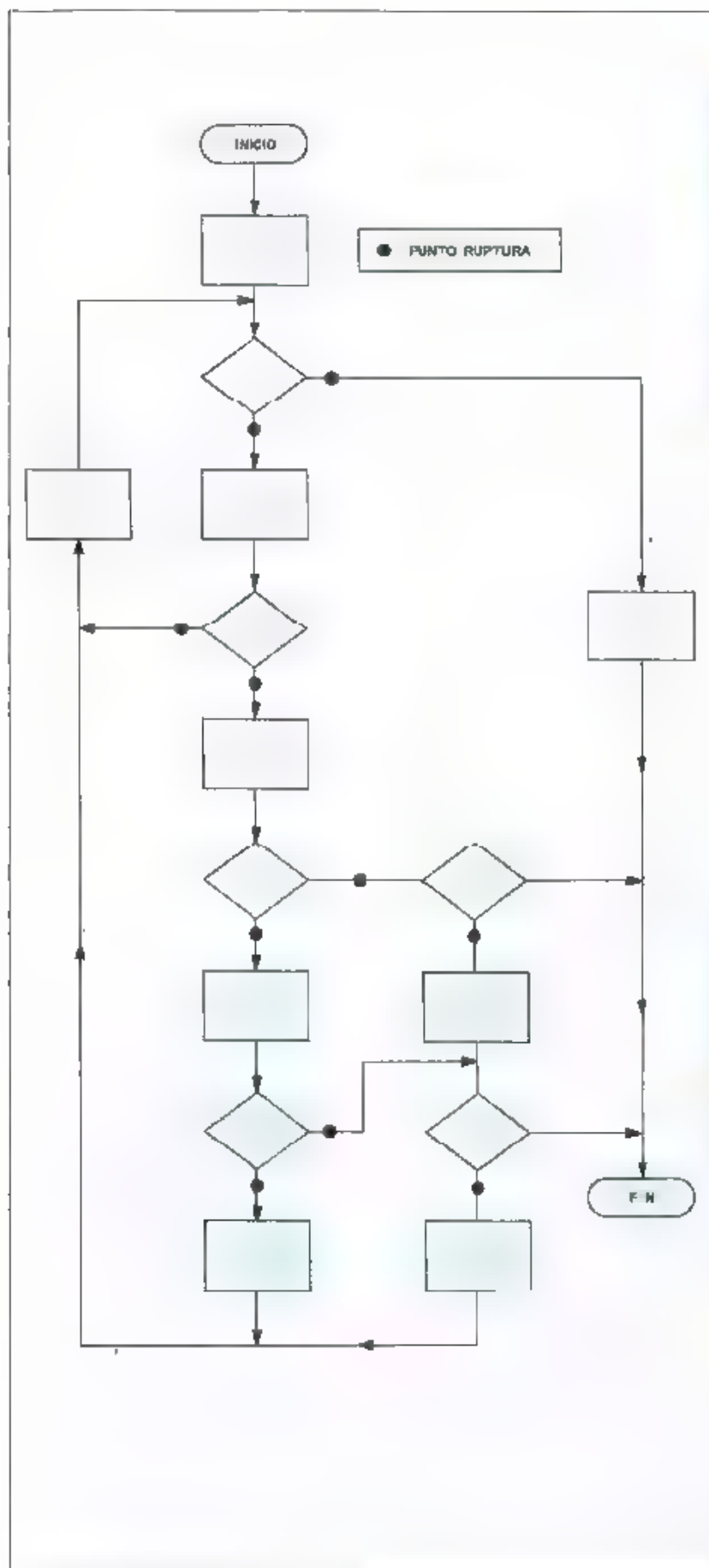
«Depurado por partes».

Los bucles anidados «p» y «n» (líneas 140-200 y 160-190) son los encargados de asignar a la matriz «T\$» los valores de cadena especificados en las sentencias «DATA» (líneas 251 a 260)

En los juegos de naipes, cada carta tiene un valor relativo distinto, en el caso de las siete y media, las cartas de «as» al «siete» tienen un valor

idéntico al que figura en una de sus esquinas. Por el contrario, las «figuras» (sota, caballo y rey) tienen un valor de medio punto. Con estos requisitos, los bucles 310-330 y 340-360 asignan a los elementos de la matriz «V» el valor correspondiente

«V» es una matriz unidimensional de diez elementos, uno por cada valor de carta. La re-



Puntos estratégicos de ruptura.

lación entre el subíndice y la carta es el siguiente:

- | | |
|----|---------|
| 1 | AS |
| 2 | DOS |
| 3 | TRES |
| 4 | CUATRO |
| 5 | CINCO |
| 6 | SEIS |
| 7 | SETE |
| 8 | OCHO |
| 9 | CABALLO |
| 10 | REY |

Para barajar las cartas se utiliza la función pseudoaleatoria «RND». La subrutina localizada en la línea 490, retorna dos valores aleatorios asignados a las variables «numero» y «palo». La variable «numero» tiene un valor comprendido entre «1» y «10»; la correspondencia, con los números de carta, es idéntica a la utilizada por la matriz «V». La variable «palo» puede tener valores entre «1» y «4», la relación entre dicho valor y el palo de la carta es:

- | | |
|---|---------|
| 1 | DIOS |
| 2 | COPAS |
| 3 | ESPADAS |
| 4 | BASTOS |

Las cartas generadas aleatoriamente son asignadas sucesivamente a los elementos de la matriz «M». La primera dimensión indica la posición de la carta dentro de la baraja, y la segunda, el número y el palo.

Al utilizar una función aleatoria pueden repetirse varios valores; para que la matriz «M» no contenga varias cartas idénticas, se ha utilizado la matriz «B». Esta matriz bidimensional de 10 por 4 está inicializada en un principio a cero.

Los valores proporciona

dos por la subrutina de generación de cartas aleatorias son utilizados como subíndices de la matriz «B», si el elemento direccionado tiene valor «0», es que la carta no se ha generado anteriormente; si, por el contrario, tiene valor «1», indica que la carta se repite. En el primer caso, los valores de las variables «numero» y «palo» se asignan al elemento correspondiente de la matriz «M» y el valor «1» al elemento direccionado de la matriz «B». En el segundo caso, no se asigna ningún valor a la matriz «M» y el programa genera dos nuevos valores que vuelven a ser comprobados.

La rutina encargada de «cortar» la baraja se encuentra localizada en las líneas 610 a 675. En la matriz «C» quedan las cartas ordenadas y listas para poder comenzar la partida.

La transferencia de cartas entre la matriz «M» y «C» se realiza en dos fases. En la primera, se transfieren los elementos situados en las primeras posiciones, es decir, desde el número uno hasta el direccionado por la variable «numero»; en la segunda, se realiza la transferencia de las restantes. En la matriz «C» estos dos bloques quedan inter cambiados.

La estructura general del programa es:

- 10 : Comenzando con el nombre del programa.
- 15 : Asignación del color azul para el borde verde para el fondo y negro para los caracteres.
- 20-26 : Carátula de programa.
- 40-75 : Asignación de valores a las variables «red tor» y «manos». Esta última contiene el número de partidas jugadas.

- 90-125
- 140-200
- 251-260
- 310-360
- 365
- 372
- 380-400
- 410-470
- 490-520
- 540-570
- 672
- 574-578
- 580
- 590
- 592-594
- 600
- 610-675
- 680-690
- 700-715
- 722
- 730
- 740
- 750
- 770

- Dimensio nado de los mar
- genes explicadas a terno
- mente.
- Bloque de asignación de
- los nombres de las cartas
- a la matriz «T».
- Datos relativos a los nom
- bros de las cartas.
- Blucías para la asignación
- de valores.
- Llamada a la subrutina de
- visualización de instrucc
- iones.
- Inicialización de la función
- aleatoria.
- Mensaje de espera.
- Asignación aleatoria de
- valores, de carta a la ma
- triz «M».
- Generación aleatoria de
- «numero» y «palo».
- Introducción del número
- de carta por donde se de
- sea cortar la baraja.
- Comprobación de que el
- valor introducido no es
- una cadena vacía.
- Bucle para detectar si hay
- algún valor no numérico
- dentro de la variable «N».
- Eliminación de la cadena
- «N».
- Comprobación de que el
- valor está dentro de los
- márgenes (1 a «40»).
- Mensaje de espera.
- Cálculo de número de
- cartas que hay desde la
- «cortada» hasta el final.
- Transferencia de valores
- entre las matrices «M» y «C»
- (cortar la baraja).
- Pantalla de «ánimo».
- Temporización hasta que
- se pulsa una tecla y borrar
- do de la pantalla.
- Invitación a jugar.
- Inicialización de los tantos
- del jugador.
- Asignación de la primera
- carta del jugador.
- Llamada a la subrutina de
- presentación de cartas.
- Asignación de los tantos

780

800-850

870

880-920

930

932

934

950

960

970

980

990

1000-1020

1030

1035

1035

1040-1050

1080-1090

1100

1110-1120

1150-1170

1175

1180-1190

- de jugador.
- Asignación de la segunda
- carta al ordenador.
- Presentación de la carta y
- asignación de su valor.
- Invitación a pedir otra carta.
- Comprobación de la res
- puesta.
- Borrar el mensaje de la zo
- na inferior.
- Temporización entre dos
- jugadas.
- Asignación de la siguiente
- carta al jugador.
- Llamada a la subrutina de
- presentación de cartas.
- Incrementar los tantos de
- jugador.
- Comprueba si el jugador
- se ha pasado.
- Comprueba si el jugador
- tiene «7¹/₂».
- Salto a mensaje de invita
- ción.
- Mensaje de presentación
- de los tantos obtenidos
- por el jugador en el caso
- de «pasarse».
- Asignación a la variable
- «ganador» del valor «0», es
- te indica que el ordenador
- gana.
- Inicialización de los tantos
- de ordenador para poder
- visualizar «7».
- Temporización hasta que
- se pulsa una tecla.
- Mensaje de obtención, por
- parte del jugador, de
- «7¹/₂».
- Indica cuál es la siguiente
- carta que debe «pedir» el
- ordenador.
- Temporiza hasta que se
- pulsa una tecla.
- Mensaje de visualización
- de los puntos obtenidos
- por el jugador en el caso
- de «quedarse».
- Borrar el mensaje de la zo
- na inferior.
- Temporización hasta que
- se pulsa una tecla.

1220	Iniciación de los tantos del ordenador	1470	Si el jugador se ha quedado p de cartas mientras que el tenga menos de seis puntos	1740	Se pulsa una tecla
1230	Asigna la segunda carta a ordenador				Si el jugador gana y tiene $(7^{1/2})$, la apuesta se duplica
1240	Indicación de que juega el ordenador	1490	Mensaje pidiendo otra carta	1750-1770	Cálculo de los nuevos créditos si gana el ordenador
1252	Coordenada (y) de la siguiente carta a visualizar	1492	Duración del mensaje	1790-1800	Idem. si gana el ordenador
1260	Llamada a la subrutina de presentación de cartas	1494	Borrado del mensaje	1820	Visualización de las (manos) jugadas hasta ese momento
1270	Asignación de los tantos del ordenador	1500	Temporización entre dos jugadas	1830	Visualización de nuevo crédito del jugador
1280	Asignación de la siguiente carta a escoger	1522	Incrementar en dos la siguiente posición	1840	Idem. del ordenador
1300	Comprueba si el ordenador se ha pasado	1530	Llamada a la subrutina de presentación de cartas	1850	Comprueba si el ordenador se ha quedado sin crédito
1310	Comprueba si el ordenador ha obtenido $(7^{1/2})$	1540	Incrementar los tantos del ordenador	1860	Idem. de ordenador
1320	Comprueba si no se ha pasado	1570-1590	Mensaje de presentación de los tantos del ordenador en el caso de quedarse	1880	Mensaje de invitación a jugar otra mano
1330	Asignación de valor (3) a la variable (ganador). El ordenador pierde.	1592-1594	Temporiza hasta que se pulsa una tecla	1890-1930	Comprobación de la respuesta elegida
1340-1360	Mensaje de presentación de los tantos del ordenador, en el caso de pasarse	1610	Si el ordenador tiene más puntos o igual que el jugador el ordenador gana.	1940	Borrado del mensaje de la zona inferior
1370-1380	Temporiza hasta que se pulsa una tecla	1620	Si tiene menos, el ordenador pierde	1945	Incremento de las (manos) jugadas
1410	Asignación del valor (0) a la variable (ganador). El ordenador gana	1640	Mensaje de que el ordenador gana.	1950-1990	Inicialización a cero de los elementos de la matriz (B para volver a barajar las cartas
1420	Mensaje de visualización $(7^{1/2})$, por parte del ordenador	1650	Mensaje de que el jugador gana.	2000	Salto a la rutina de barajar
1430-1457	Temporiza hasta que se pulsa una tecla.	1660	Visualización de los tantos del ordenador	2100	Comienzo de la subrutina INSTRUCCIONES.
1460	Comprueba si el jugador ha obtenido $(7^{1/2})$, para seguir pidiendo cartas hasta que gane o se pase	1680	Si el jugador se quedó, el ordenador no indica cuál era su carta	2110	Invitación a visualizar las instrucciones.
		1690	Visualización de los tantos del ordenador	2120-2160	Comprobación de la operación elegida
		1700-1710	Temporización hasta que	2170-2210	Visualización de las instrucciones.
				2220-2230	Pausa hasta que se pulsa una tecla

DEPURACION DE PROGRAMAS

En capítulos anteriores se ha ido explicando la mayor parte del juego de sentencias del Spectrum, así como una serie de conceptos: Bucles, Subrutinas, Funciones, Matrices, etc. Con todo este material el iniciado en programación puede ya confeccionar un volumen importante de programas; por este motivo es necesario que conozca una serie de técnicas destinadas a la depuración o puesta a punto de programas; también son conocidas por el término inglés *debugging*.

Errores

Es muy difícil que un programa, medianamente complicado funcione a la primera, ya que hay muchos factores que deben tenerse en cuenta.

Los tipos de error que surgen al elaborar un programa, pueden clasificarse, principalmente, en tres apartados:

SINTACTICOS
DE DEFINICION
DE ESTRUCTURA

Los errores *sintácticos* son aquellos que se producen al teclear un argumento no compatible con una sentencia determinada, por ejemplo:

```
- NEW 32  
- LET "A$" = "abc"  
- CLS fin  
- PRINT $
```



Zonas de la pantalla.

Este tipo de errores son detectados inmediatamente por el intérprete BASIC, al pulsar «ENTER». Una interrogación parpadeante se posiciona en las proximidades del error, indicando que no entiende dicha sentencia.

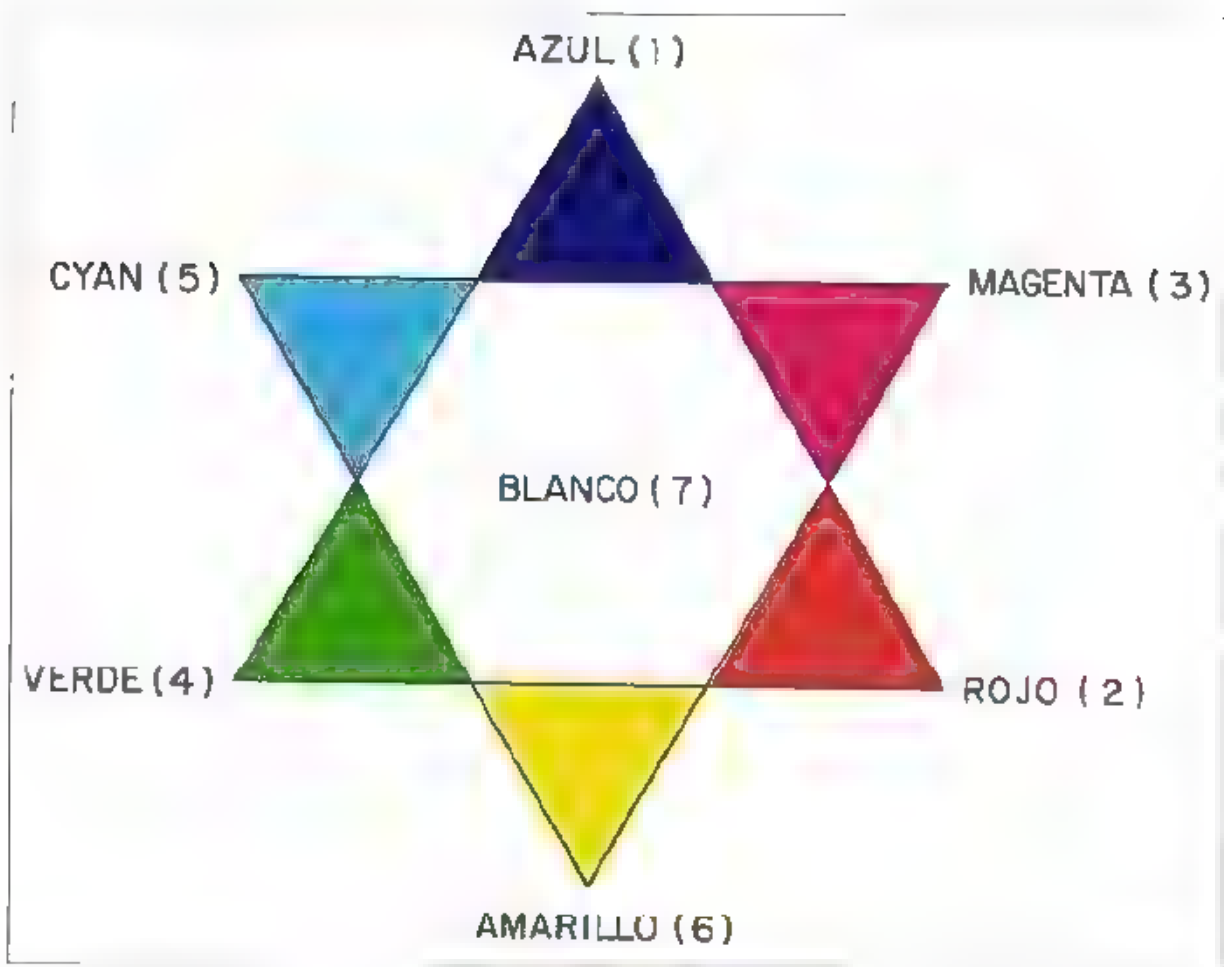
Revise la línea introducida; si no encuentra el fallo, consulte, en el capítulo correspondiente del libro, las posibles estructuras que puede adoptar dicha sentencia.

Como puede observar, los errores sintácticos son detectados y corregidos en la

fase de edición de programas, a diferencia de otros ordenadores en los que se realiza posteriormente.

Otro tipo de errores se manifiestan al ejecutarse el programa, presentando el correspondiente mensaje de error. Estos errores son originados principalmente por no haber definido previamente una variable, por utilizar un subíndice fuera de rango, por utilizar un parámetro erróneo en una función, etc.

El mensaje presentado sirve únicamente de referencia



Estrella de colores.

para encontrar el error, ya que indica el tipo y la línea en que lo ha detectado (ver pag. 96). El número de línea no siempre significa que el error se encuentre en la misma, sino que está relacionado con ella, por ejemplo, un error con mensaje:

```
2 Variable not found 70 1
```

suponiendo que la línea 70 fuera:

```
70 PRINT mes
```

significa que el argumento de «PRINT» al no ir entrecomillado, lo interpreta como variable y no encuentra su asignación inicial. Si efectivamente

«mes» era el nombre de una variable, solucionaríamos el problema asignando a esta un valor en la zona del programa que correspondiera, por ejemplo:

```
25 LET mes = 8
```

Si «mes» no fuera una variable, sino que por el contrario fuera una cadena a visualizar el error se encontraría en la misma línea 70, ya que debería ser:

```
70 PRINT "mes"
```

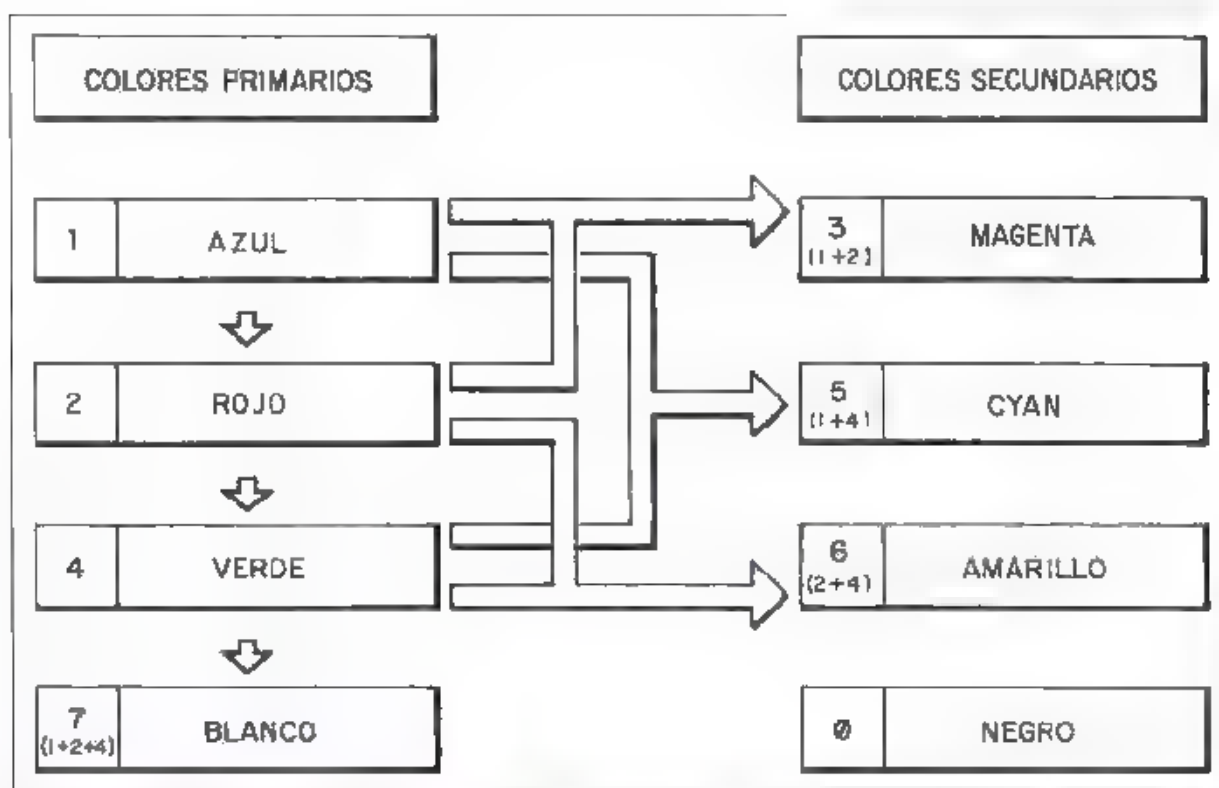
Los errores más difíciles de localizar son aquellos que no generan ningún tipo de mensaje de error, pero que hacen

que nuestro programa no realice las funciones que en un principio teníamos previstas. Al corregir este tipo de errores, de *estructura*, están concebidas las técnicas de depuración.

Depuración

La tarea de depurar no es muy complicada, pero sí laboriosa, aunque a veces basta con echar una ojeada al *listado*, para descubrir el error.

La complejidad de la depuración depende del número de bifurcaciones que tenga el programa, ya que puede ser bastante elevado el número de caminos distintos que siga



Síntesis aditiva de colores.

al ejecutarse.

Tres cosas que facilitan bastante la depuración son:

- DIAGRAMAS DE FLUJO
- LISTADOS
- USO DE SUBROUTINAS

El tener confeccionado el diagrama de flujos nos permite ver las funciones que realiza el programa paso a paso y los diversos caminos que debe seguir según se cumplan ciertas condiciones. Podemos localizar, con su ayuda,

la zona o el bloque de instrucciones donde falle el programa.

Si se dispone de una impresora, podemos obtener un listado y así localizar con mayor facilidad las sentencias que consideremos oportunas.

El uso de subrutinas facilita la depuración, ya que éstas pueden ser comprobadas individualmente, de esta manera se pueden tener ciertas garantías de que funcionan cuando se depura el programa final.

Es importante tener localizada la zona a partir de la cual se desea comenzar la depuración por ejemplo, en el siguiente programa, que tiene dos errores, empezariamos a depurar a partir de la línea 40, ya que si lo ejecuta observará que se visualizan las constantes de cadena de las líneas 20 y 30.

Para corregir el ejemplo, deberá modificar la línea 60.

```
60 PRINT AT n + 20, a$
```

PROGRAMA 1

```
10 REM *****
   *          *
   * ERRORES *
   *          *
   * XXXX.XXXX
20 PRINT ARTICULO, "PRECIO,"
30 PRINT
```

```
40 FOR n=1 TO 5
50 INPUT "Artículo > ",a$
60 PRINT AT 20,a$
70 INPUT "Precio > ",p
80 GET
90 PRINT AT n+2 16,p
```


e introducir la línea 90 dentro del bucle, asignándola un nuevo número de línea comprendido entre 71 y 79

Lamentablemente el Spectrum no dispone de dos sentencias muy potentes utilizadas en otros ordenadores para depurar. Estas son

- TRON
- TROFF

que permiten habilitar o deshabilitar la facilidad de traza, con ella se puede visualizar en pantalla la secuencia de instrucciones ejecutadas por el ordenador, paso a paso y, comprobar si es correcta

El Spectrum utiliza para depurar las sentencias «STOP» y «CONTINUE», utilizando la primera como *breakpoint* o punto de ruptura.

STOP y CONTINUE

La utilización de estas sentencias es bastante sencilla, pero si tiene alguna duda consulte las páginas 91 a 96.

Básicamente la depuración

con estas sentencias consiste en colocar en lugares estratégicos del programa, diversos puntos de ruptura con la sentencia «STOP».

Al ejecutarse deberá pararse en el primer punto, visualizando el manejo correspondiente, si no lo encuentras es que el fallo está localizado entre la línea de arranque y dicho «STOP». Si por el contrario se para, podremos mediante comandos directos conocer el valor de las variables utilizadas; si su contenido es correcto introduciremos también como comando directo la sentencia «CONTINUE» y el programa continuará su ejecución hasta el próximo punto de ruptura, donde haremos las mismas operaciones.

De esta manera iremos ejecutando por partes el programa hasta que localicemos el fallo

Algunos de los puntos estratégicos para la colocación de los «STOP» son las bifurcaciones, es decir, donde el programa pregunta si se cumple una condición im-

puesta en un «IF ... THEN ...».

Una vez parado un programa si se desea continuar con su ejecución, en una línea determinada, debe utilizar «GO TO» o «GO SUB», si se trata de una subrutina, ya que si utiliza «RUN n» todas las variables que tenía definidas se borran y posiblemente aparezcan el error:

2 Variable not found

por que la definición se encontraba en las líneas anteriores.

Cuando el programa esté corregido deben suprimirse la líneas con las sentencias «STOP», utilizadas como puntos de ruptura.

Programa «Depurador»

Como alternativa al uso «STOP» y «CONTINUE» el programa «1» realiza las mismas funciones, pero tiene la ventaja de una mayor facilidad de uso

El programa que usted quiera depurar deberá estar

PROGRAMA 2

```

10 REM *****
  *  CURSO/BASIC  *
  *  *****  *
  *  11  ERRORES  *
  *  *****  *
20 BORDER 4 PAPER 4 INK 0 C
LS
30 REM  VARIABLES, LET logo=1
000
35 LET igual=1100
40 GO TO logo
50 INPUT "Cadena 1 >>> ",a$
60 IF a$="" THEN GO TO 1000
70 IF LEN a$ 20 THEN GO TO 50
80 PRINT AT 3 0, "Cadena 1"
$
90 INPUT "Cadena 2 >>> ",b$
100 IF b$="" THEN GO TO 90
110 IF LEN b$ 20 THEN GO TO 90
120 PRINT AT 10,0, "Cadena 2"
*
130 IF a$=b$ THEN GO SUB igual
140 GO SUB distinto

```

```

150 REM CONTINUAR
160 PRINT 80, "Desea continuar (
S/N)"
170 PAUSE 0
180 LET z$=INKEY$
190 IF z$="S" OR z$="s" THEN GO
T 190
200 IF z$="N" OR z$="n" THEN CL
S PAUSE 0 STOP
210 GO TO 170
1000 REM LOGO
1005 RESTORE
1010 FOR n=1 TO 32
1020 READ d310
1030 PRINT INVERSE 1,AT 0,n,CHR$
dato
1040 NEXT n
1050 DATA 32,7,73,90,82,73,72,7
9,66,66,89,32,33,69,74,85,76,66
76,32,32,32,32,32,32,32,32,32
1060 RETURN
1100 PRINT AT 7,0, "Cadenas #1: y
#2: distinto"
1110 RETURN
1200 PRINT AT 7,0, "Cadenas #1: y
#2: iguales"
1210 RETURN

```

PROGRAMA 3

```

10 REM *****
  * CLASO BASI. *
  * ***** *
  * SIN ERRORES *
  * ***** *

20 BORDER 4 PAPER 4 INK 0 0
L3
30 REM VARIABLES
32 LET logo=1000
35 LET distinto 1100
37 LET igual 1200
40 GO SUB logo
50 INPUT "Cadena 1" ,a$
60 IF a$="" THEN GO TO 50
70 IF LEN(a$)=20 THEN SUB TO 50
80 PRINT AT 3,0,"Cadena 1" ,a$

90 INPUT "Cadena 2" ,b$
100 IF b$="" THEN GO TO 90
110 IF LEN(b$)=20 THEN GO TO 90
120 PRINT AT 5,0,"Cadena 2" ,b$

130 IF a$=b$ THEN GO SUB igual
GO TO 150

```

```

140 GO SUB distinto
150 REM CONTINUAR
160 PRINT #0,"Cesó continuar"
S/N
170 PAUSE 0
180 LET z$=TN*FN$
190 IF z$="S" OR z$="s" THEN GO TO 10
200 IF z$="N" OR z$="n" THEN CL
PAUSE 0 STOP
210 GO TO 170
1000 REM LOGO
1005 RESTORE
1010 FOR n=0 TO 31
1020 READ dato
1030 PRINT INVERSE 1,AT 0,n,CHR$
dato
1040 NEXT n
1050 DATA 32,77,73,67,82,79,72,
0,66,62,80,76,60,69,77,64,70,60,
76,32,32,32,32,32,32,32,32,32,
32,32,32
1060 RETURN
1100 PRINT AT 7,0,"Cadenas #1 y
#2: distintas"
1110 RETURN
1200 PRINT AT 7,0,"Cadenas #1 y
#2: iguales"
1210 RETURN

```

ubicado en las líneas de «1» a la «9978», para no crear conflicto. Suponiendo que al saberlo le haya llamado «debug» podrá *combinar* ambos programas de la siguiente manera:

- Deberá cargar primero el que quiera depurar, si no lo tiene en memoria.
- Para cargar el programa «debug» utilice.

MERGE debug

Cuando los dos estén en memoria, inserte el siguiente comando directo para facilitar su manejo.

LET debug = 9980

Para colocar los puntos de ruptura utilice, en lugar de «STOP», la sentencia

GO SUB debug

Una vez ejecutado el programa a depurar, cuando encuentra un punto de ruptura se ejecuta la subrutina «debug» que presenta el mensaje

Parada en

indicando la línea y la sentencia dentro de la línea.

Si se pulsa cualquier tecla menos la «V» el programa continúa hasta el próximo punto de ruptura. Por el contrario, si se pulsa la «V», la subrutina nos pregunta que variable deseamos visualizar:

Variable >

introduciendo el nombre y pulsando «ENTER» nos aparece su contenido.

Pulsando cualquier tecla nos vuelve a preguntar el nombre de otra variable, si no deseamos conocer el conte

nido de ninguna más, basta con pulsar «ENTER» para que el programa principal continúe hasta el próximo punto de ruptura.

Cuando el programa esté depurado deberá eliminar todos los puntos de ruptura así como las líneas «9970» a «9999».

Ejercicio

Como programa-ejercicio de este capítulo, edite el número 2. Se encuentra, lógicamente libre de errores *sintácticos*, pero se han introducido «11» errores de *definición* y de *estructura*. Intente, con la ayuda de las técnicas explicadas, localizar y corregir dichos errores.

Si se encuentra «desesperado» o desea comparar sus resultados, el programa número 3 ofrece una de las posibles soluciones.

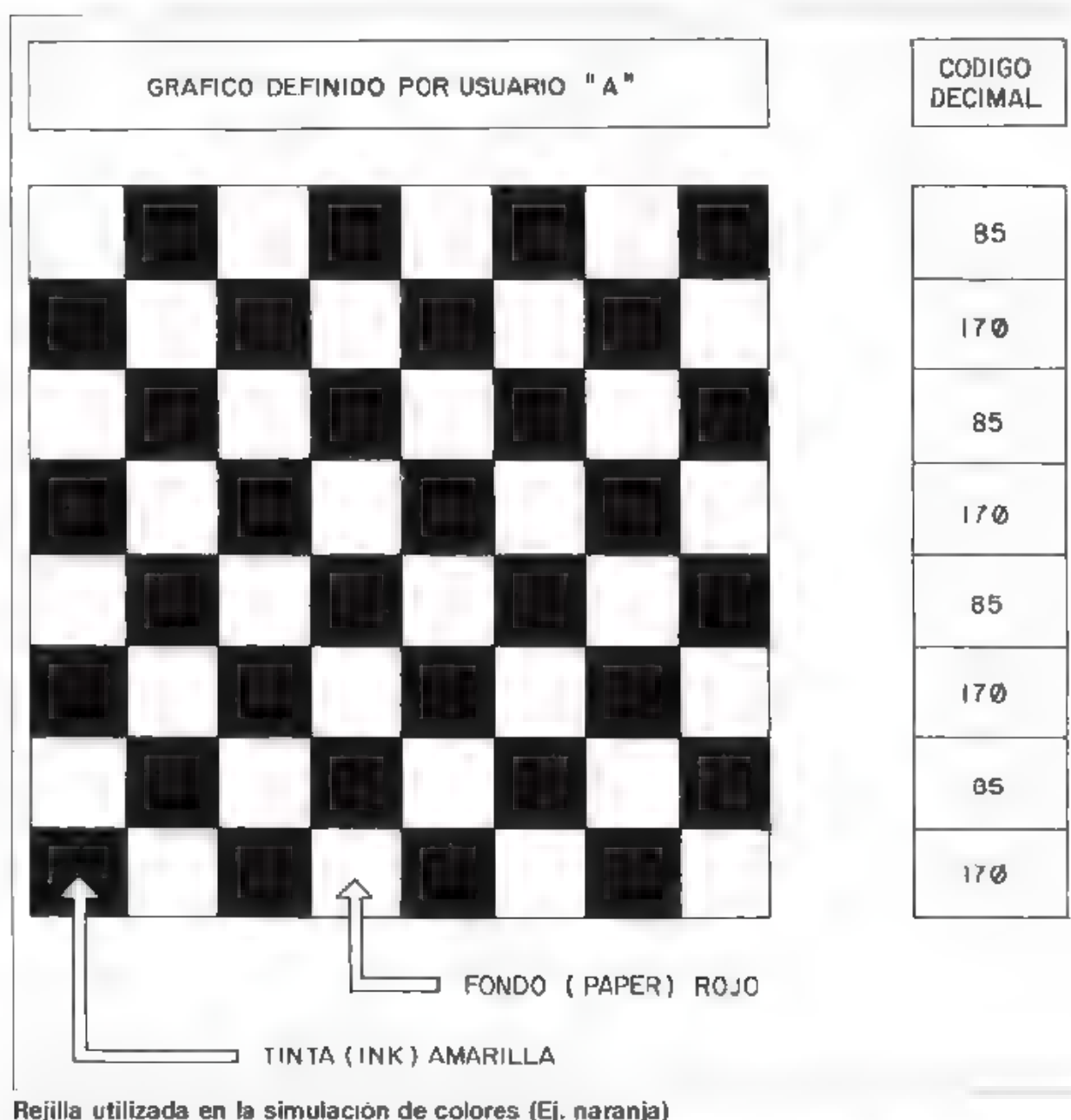
COLOR

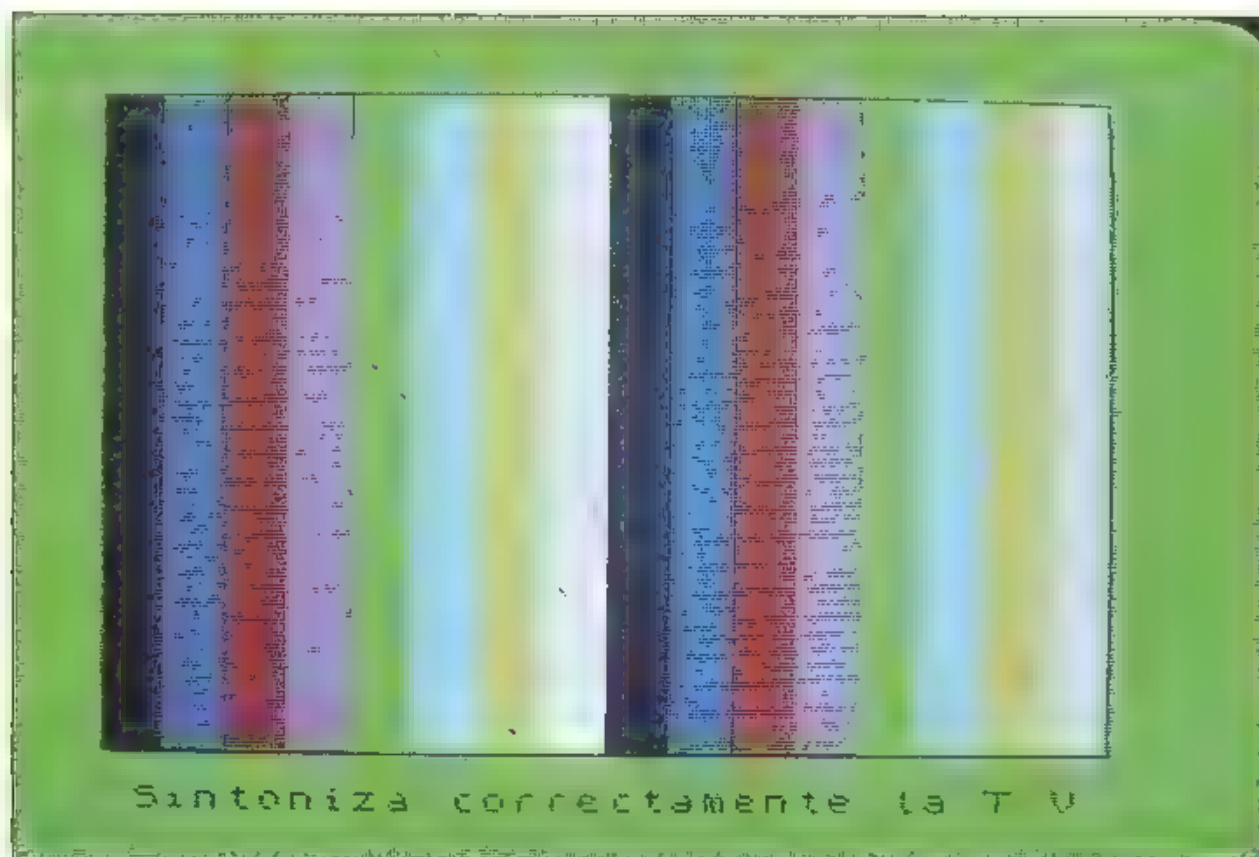
El color es actualmente una de las características a tener en cuenta cuando se adquiere un ordenador personal, principalmente si lo piensa

destinar a programas de juegos, aunque en otro tipo de aplicaciones tampoco está de más añadir algo de color, bien sea por estética o para

destacar algún mensaje o zona de pantalla

El «ZX Spectrum», hermano mayor del conocido «ZX-81» debe precisamente su nombre a la capacidad que tiene





Carta de ajuste

para generar los colores del espectro luminoso.

El Spectrum dispone de ocho colores que pueden conseguirse en dos gamas de brillo; cada uno tiene asignado un número que lo identifica a la hora de programar

CODIGO	COLOR
0	NEGRO
1	AZUL
2	ROJO
3	MAGENTA
4	VERDE
5	CYAN
6	AMARILLO
7	BLANCO

Este código está en función de la luminosidad del color, así el negro o ausencia de luz, tiene el código cero y según va progresando la luminosidad aumenta el valor hasta llegar al blanco que tiene el

código siete. La luminosidad de los colores es difícil de apreciar ya que el ojo humano no es más sensible a ciertos colores, los cuales, nos parecen más claros

El programa n.º «1» muestra unas barras verticales con diversos colores, éstos están ordenados según su código. Si dispone de un televisor en blanco y negro podrá comprobar la luminosidad de cada color ya que cada uno toma un valor distinto de gris; el resultado es una escala degradada de grises. Obtendrá el mismo efecto en un televisor de color, si éste es anulado; el programa también sirve para poder sintonizar correctamente el televisor

Teoría del color

Antes de explicar la utiliza-

ción de los comandos que afectan sobre el color de las diversas zonas de la pantalla, es conveniente tener algunos conocimientos sobre la teoría del color.

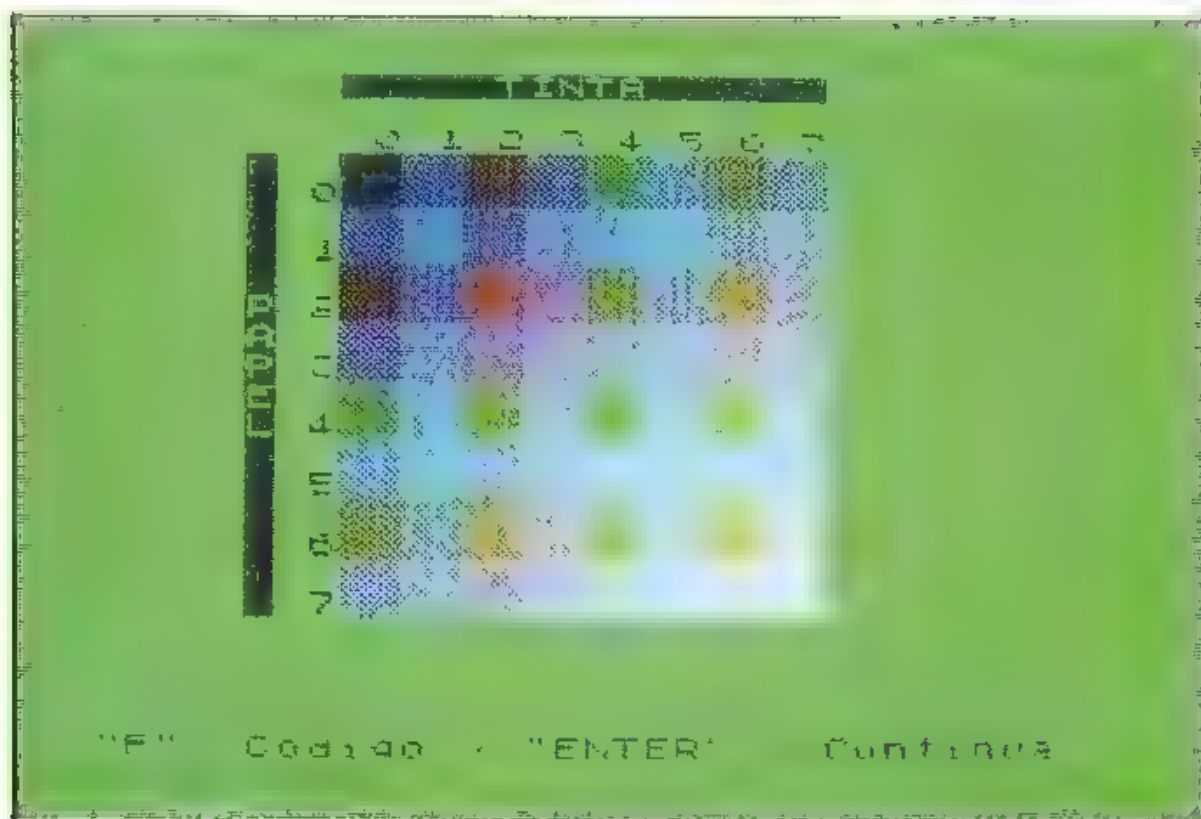
La descomposición de la luz blanca, por un prisma, se reduce básicamente a tres colores.

AZUL
- ROJO
VERDE

éstos son llamados «primarios». A partir de éstos, y por *síntesis aditiva*, pueden obtenerse los llamados «secundarios».

- MAGENTA
- CYAN
- AMARILLO

Se denominan colores



«64» Colores

complementarios aquellos que al mezclarse, por síntesis aditiva, dan origen, de nuevo, a la luz blanca. En la estrella de los colores, representada en la figura, pueden observarse los colores complementarios, ya que éstos se encuentran enfrentados.

AZUL	AMARILLO
ROJO	CYAN
VERDE	MAGENTA

Lógicamente, al recomponer los tres colores primarios se obtiene también la luz blanca.

Síntesis aditiva

Esta se demuestra con la utilización de tres proyectores de luz blanca. En el primero insertamos un filtro de color

rojo, en el segundo uno verde y en el tercero uno azul.

Los proyectores se colocan de manera que sus haces coincidan según la figura adjunta.

Se comprueba que en el área iluminada por los tres proyectores se obtiene luz blanca, en el área de intersección de la luz roja con la verde, el amarillo, en el caso de la luz roja y del azul, resulta el magenta y por último, de la mezcla de la luz verde y azul, de cyan.

ROJO + VERDE + AZUL	BLANCO
ROJO + VERDE	AMARILLO
ROJO + A.Z.	MAGENTA
VERDE + AZUL	CYAN

Comprobemos que en el Spectrum se cumple la síntesis aditiva, sumando los respectivos códigos de color.

a) Blanco

Azul	1
Rojo	2
Verde	4
Blanco	7

b) Magenta

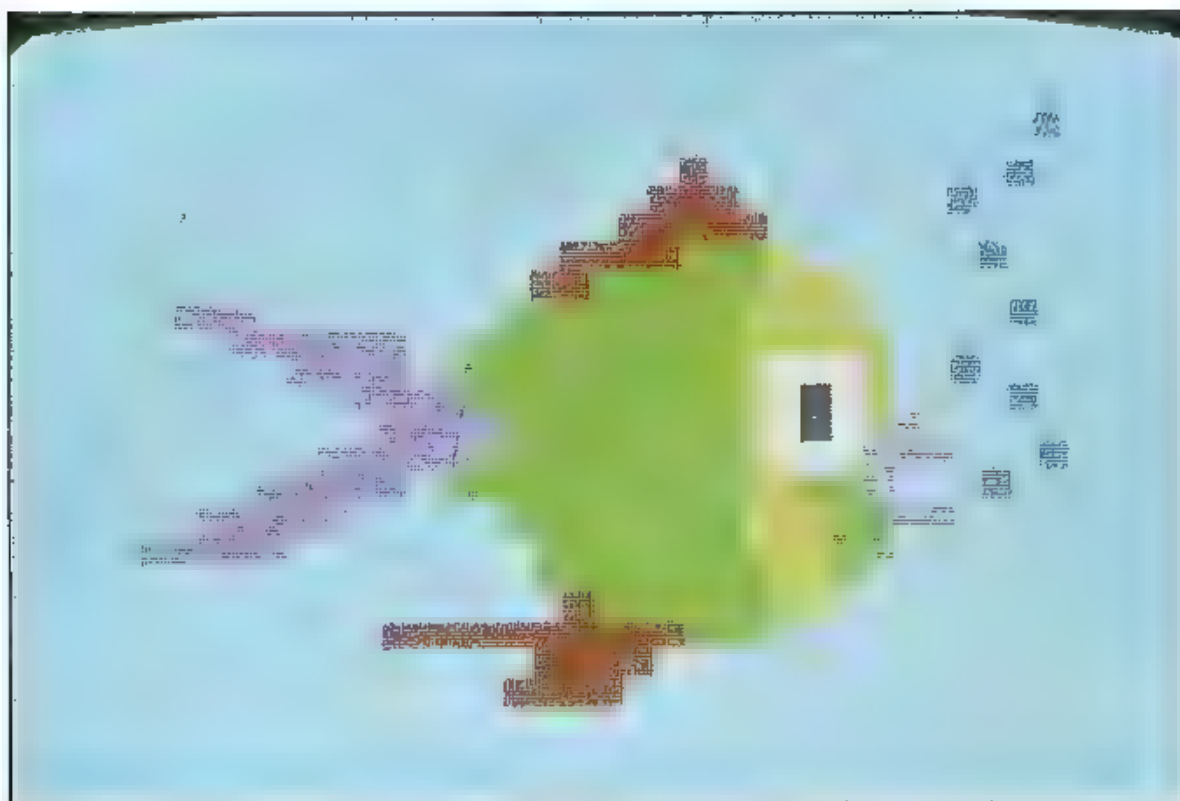
Azul	
Rojo	2
Magenta	3

c) Cyan

Azul	1
Verde	4
Cyan	5

d) Amarillo

Rojo	2
Verde	4
Amarillo	6



Las posibilidades del color

Zonas de pantalla

Existen dos zonas que pueden ser modificadas por los comandos de color de color

BORDE
PAPEL

El *borde*, como su propio nombre indica, es la zona formada por el margen exterior de la pantalla del televisor, en éste no se pueden imprimir caracteres.

El *papel* o fondo es el rectángulo central, rodeado por el «borde» donde se imprimen los caracteres o gráficos.

Las siguientes instrucciones muestran estas zonas



El mensaje «BORDE rojo» está visualizado en la línea 23 (zona reservada), ésta al igual que la zona 22 toman el color del «borde».

El color en que visualizamos los caracteres se denomina *tinta*.

Comandos



Acceso al teclado

BIN



MODO

BRIGHT

Definición

Con este comando puede alterarse el color del borde. La estructura general es

SENTENCIA	ARGUMENTO
BORDE	color (0-15)

Ejemplos

- BORDER 2 (rojo)
- BORDER 3 (magenta)
- BORDER 0 (negro)
- BORDER 7 (blanco)

Observará que nada más introducir el comando, el borde cambia de color.

El siguiente programa cambia el color del borde cuando se pulsa una tecla, si la mantiene constantemente pulsada éste cambia de color a toda velocidad.

```
10 REM *****
   : BORDER :
   : *****

20 RESTORE
30 READ A$(0 OF
35 IF A$="FIN" THEN GO TO 20
40 BORDER COLO
45 CLS
50 PRINT "B" AT 1,10 BORDER
55 PRN
55 IF INKEY$="" THEN GO TO 50
55 GO TO 30
70 DATA negro 0 azul 1 rojo
80 2 verde 3 blanco 4
90 5 negro 6 azul 7 rojo
100 8 verde 9 blanco
```

PAPER

Acceso al teclado

L PRINT



MODO E

PAPER

Definición

Permite modificar el color del fondo o papel.

Su estructura general es la siguiente:

SENTENCIA	ARGUMENTO
PAPER	color de fondo

Ejemplos

- PAPER 2
- PAPER «MICRO-HOBBY»
- PAPER 5
- PRINT «SEMANAL»

Cuando se introduce una sentencia de este tipo, observará que el fondo no cambia de color, para que los atributos de éste cambien, debe pulsarse dos veces consecutivas la tecla «ENTER» cuando el comando es directo, o introducir la continuación del comando «CLS», ya que de lo contrario, el fondo cambia de color únicamente en las zonas donde se imprime algo.

El siguiente programa altera, de forma parecida al anterior, el color del papel.

```
10 REM *****
   : PAPER :
   : *****

20 BORDER 7
30 RESTORE
40 FOR N=0 TO 9
50 READ A$
60 PAPER N
70 CLS
80 PRINT AT 10,5 PAPER N
90 PRN
95 IF INKEY$="" THEN GO TO 90
100 NEXT N
110 GO TO 30
120 DATA negro azul rojo
130 verde blanco
```

(N)

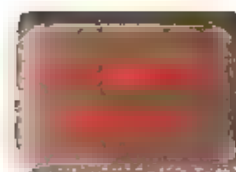
Acceso al teclado

EXP



MODO E

INK



Definición

El color de la tinta o del carácter es modificado con esta sentencia.

Su estructura general es:

ESTRUCTURA	ARGUMENTO
INK	color de tinta

Ejemplos:

- INK 0
- PRINT «MICRO»
- INK 2
- PRINT «HOBBY»

Para que los atributos cambien debe operarse de la misma manera que en el caso de PAPER.

Las siguientes instrucciones cambian secuencialmente el color de la tinta.

```
10 REM *****
   : TINTA :
   : *****

20 BORDER 7 PAPER 7 CLS
30 RESTORE
40 FOR N=0 TO 9
50 READ A$
60 INK N
70 CLS
80 PRINT AT 10,5 TINTA N
90 PRN
95 IF INKEY$="" THEN GO TO 90
100 NEXT N
110 GO TO 30
120 DATA negro azul rojo
130 verde blanco
```

Por la misma razón que en el programa anterior, el mensaje, en tinta blanca, no se visualiza.

Ejecuta el siguiente programa que combina todas las posibilidades de «borde», «papel» y «tinta».

[illegible]

Resolución del color

Cada carácter está formado por una matriz de ocho por ocho puntos, también conocidos por el término inglés «pixel», por tanto hay 64. A pesar de que podemos activar individualmente cada punto, con las sentencias utilizadas en la realización de gráficos en alta resolución, no se puede, sin embargo, asignar un color de tinta distinto para cada uno de ellos, ya que cada carácter está controlado por unos atributos, bien permanentes o temporales. Estos afectan a cada matriz de 64 pixels, por tanto sólo puede haber dos colores distintos en cada posición de carácter.

Esta configuración se denomina «color en baja resolución».

Transparencia y contraste

Las sentencias «PAPER» e «INK» pueden tener como argumento los códigos de color 8 y 9, teniendo un significado de *transparencia* y *contraste* respectivamente.

La transparencia consiste en conservar los atributos temporales de la pantalla al imprimir un nuevo carácter.

Ejemplos:

a) Papel transparente:

```
10 REM *****
11          PAPER 8
12          *****
13 BORDER 1 PAPER 2 INK 0 C
14 FOR I=1 TO 5
15   FOR J=1 TO 5
16     PRINT PAPER 4
17   NEXT J
18 NEXT I
19 PRINT AT 1,1 MICROMOBIE SE
20 PRINT AT 1,4 PAPER 5 MIC
21          DEMONSTR
```

El mensaje de la línea 70 «machaca» los atributos temporales, imprimiéndose sobre fondo verde, ya que este es el atributo permanente especificado en la línea 12.

Sin embargo, el mensaje de la línea 80 conserva estos atributos.

b) Tinta transparente

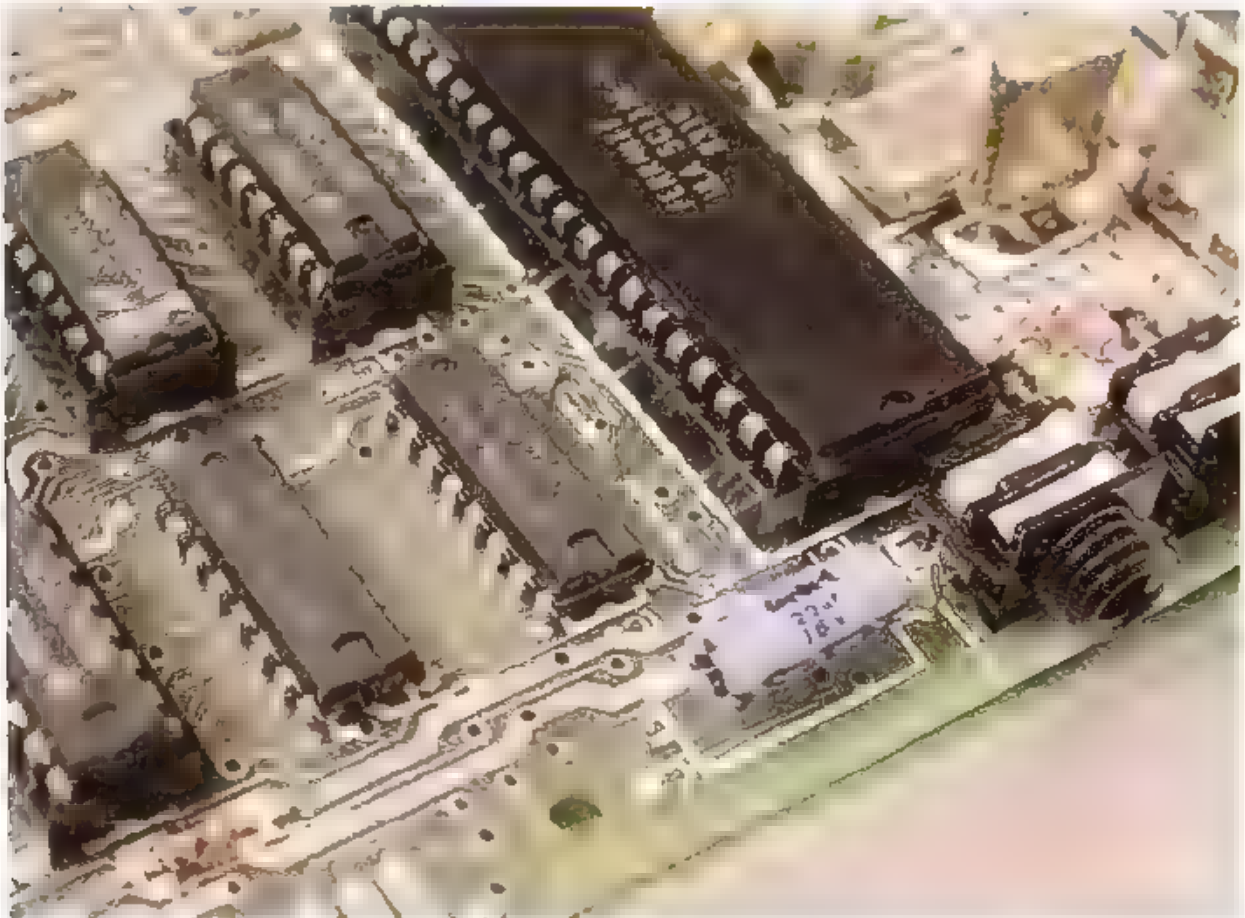
Introduzca el siguiente comando directo que selecciona los colores «rojo» para borde y fondo, y «negro» para tinta.

```
BORDER 2 PAPER 2 INK 0 CLS
```

Teclee el siguiente bucle:

```
FOR N=0 TO 7
PRINT AT 1,0, INK N; MICR0; D88V;
NEXT N
```

que visualiza la cadena «MI-



Circuito generador de colores.

CROHOBBY» con diversos colores temporales de tinta. Introduzca ahora este otro bucle

```
FOR n = 0 TO 10:
PRINT AT n 0, NK 8
MICROHOBBY SEMANA!
NEXT n
```

que visualiza la cadena «MICROHOBBY» con diversos colores temporales de tinta. Introduzca ahora este otro bucle

```
FOR n = 0 TO 10:
PRINT AT n 0, NK 8
MICROHOBBY SEMANA!
NEXT n
```

Observe como son respetados los atributos temporales de tinta

El contraste es una característica que indica que el color correspondiente a papel o tinta va a ser en blanco o negro para destacar sobre el otro

El color de contraste es «negro» cuando el otro es de tonalidad clara, y «blanco» cuando es oscura.

CONTRASTE	COLOR
	NEGRO
BLANCO	AZUL
	ROJO
	MADEIRA
	VERDE
NEGRO	CYAN
	AMARILLO
	BLANCO

El siguiente programa cambia el color de borde y papel observe el color de la tinta

```
30 REM *****
PRINT AT 0 0, CONTRASTE
*****
```

```
20 FOR n=0 TO 10:
30 BORDER n, PAPER n, CL 5
40 PRINT AT n 0, NK 8
MICROHOBBY SEMANA!
50 NEXT n
```

Tanto los mensajes que envía el ordenador como los que usted visualiza a través de los canales de comunicación cerebro y uno (0 y 1), tienen la característica de tinta de contraste, en relación con el color del borde

Edita el siguiente programa que genera con colores temporales, una serie de barras verticales

```
10 REM *****
PRINT AT 0 0, TINTA 9
*****
20 FOR n=0 TO 10:
30 BORDER n, PAPER n, CL 5
40 PRINT AT n 0, NK 8
MICROHOBBY SEMANA!
50 NEXT n
```

Los mensajes de las líneas 90 y 100 están visualizados sobre papel transparente. El primer mensaje no se puede leer correctamente debido al poco contraste que existe en algunas zonas, entre el papel y la tinta, sin embargo, el segundo aprovecha la característica de contraste de tinta para ser legible.

En este otro ejemplo se demuestra la característica de papel de contraste.

```
30 REM *****
PRINT AT 0 0, PAPER 9
*****
20 BORDER 2, PAPER 2, CL 5
30 PRINT AT 0 0, NK 8
MICROHOBBY SEMANA!
40 NEXT n
```

Con el papel de contraste, aparte de quedar el mensaje destacado, se visualiza la cadena «HOBBY»

Simulación de colores

Aparte de los ocho colores de que dispone el Spectrum pueden simularse hasta «28» más, disponibles también en dos gamas de brillo, por tanto pueden conseguirse en total hasta «72» tonalidades distintas [(28 + 8) * 2]

La forma de simularlos es bastante simple. Utilizando una rejilla pequeña, similar a un tablero de ajedrez, en que los cuadros blancos corresponden al fondo y los negros a la tinta, se observa que asignando diversos colores al papel y a la tinta, y situada la rejilla a cierta distancia, el ojo integra ambos colores dando como resultado uno distinto que es la mezcla; por ejemplo, con el rojo y el amarillo simulamos el naranja

Con el programa número «2», que utiliza los gráficos definidos, asignamos a la letra «A» una rejilla similar a la explicada

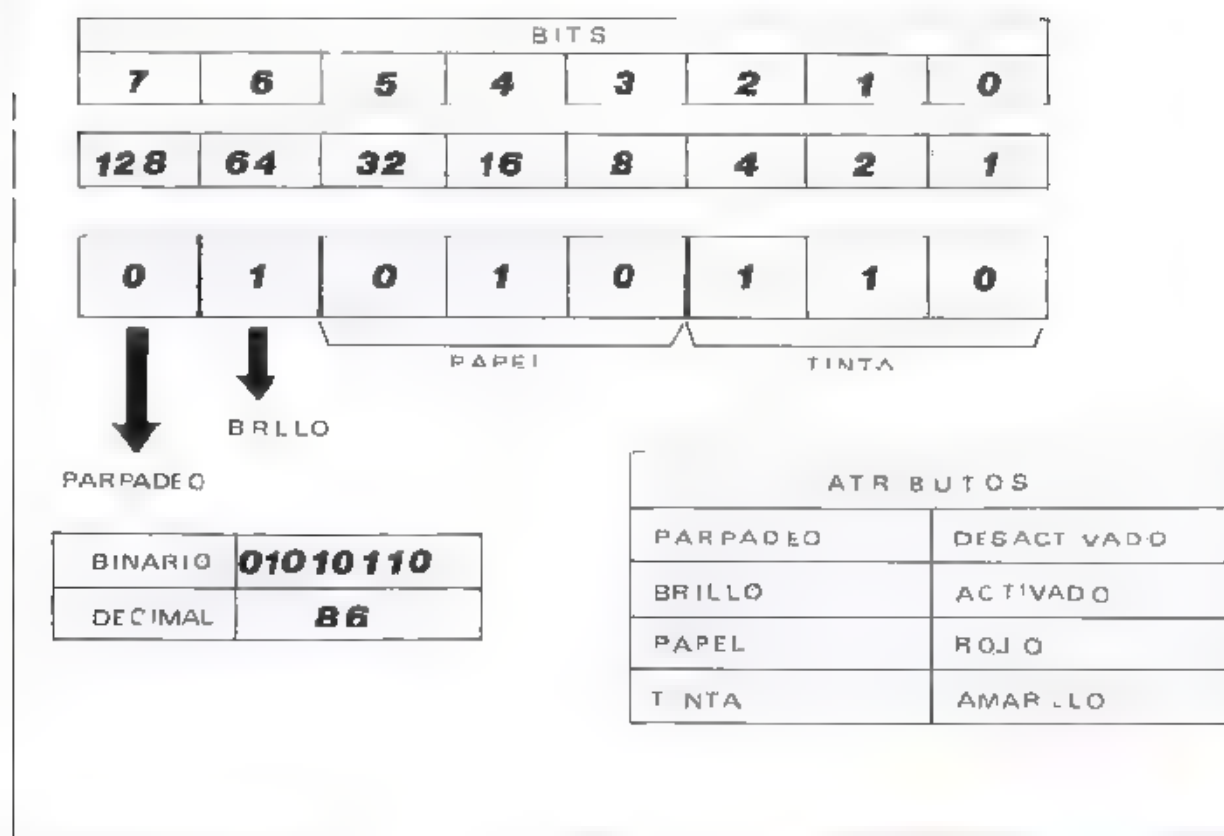
No se preocupe si al ejecutarse la línea 60 el programa desaparece, visualizándose el conocido mensaje:

© 1982 Sinclair Research Ltd

ya que el gráfico permanece en memoria podrá comprobarlo, seleccionando el modo IQ (gráficos) y pulsando la letra «A».

Retorne al modo anterior y ejecute el programa número «3».

Dicho programa nos presenta, utilizando a rejilla, todas las combinaciones de tin-



Ejemplo función «ATTR».

ta y papel. Observará que algunos colores se encuentran repetidos ya que la combinación de papel «3» y tinta «5», es la misma que papel «5» y tinta «3».

Si pulsamos cualquier tecla, los colores se visualizan en una gama de brillo distinta. Si por el contrario pulsamos «CAPS SHIFT» simultáneamente con la tecla «F», accedemos a la opción «CÓDIGO».

El código debe ser introducido de la siguiente manera

- Teclar el código del papel (0 - 7)
- Teclar el código de la tinta (0 - 7)
- Pulsar «ENTER».

Este código selecciona el color simulado por la mezcla de ambos, el cual será visualizado con su brillo correspondiente, en la zona de pantalla

conocida como fondo.

Pulsando cualquier tecla se accede otra vez al «muestreo» de colores

Control de Impresión

Con el uso de ciertos comandos se pueden alterar las siguientes características de impresión

BRILLO
INVERSIÓN DE VIDEO
PARPADEO

Estas, al igual que los colores, pueden ser permanentes o temporales.

El argumento de estas sentencias indica si se desea activar o desactivar cierta característica, utilizando para ello un código de control

CÓDIGO	SIGNIFICADO
0	DESACTIVADO
1	ACTIVADO

BRILLO

Acceso al teclado

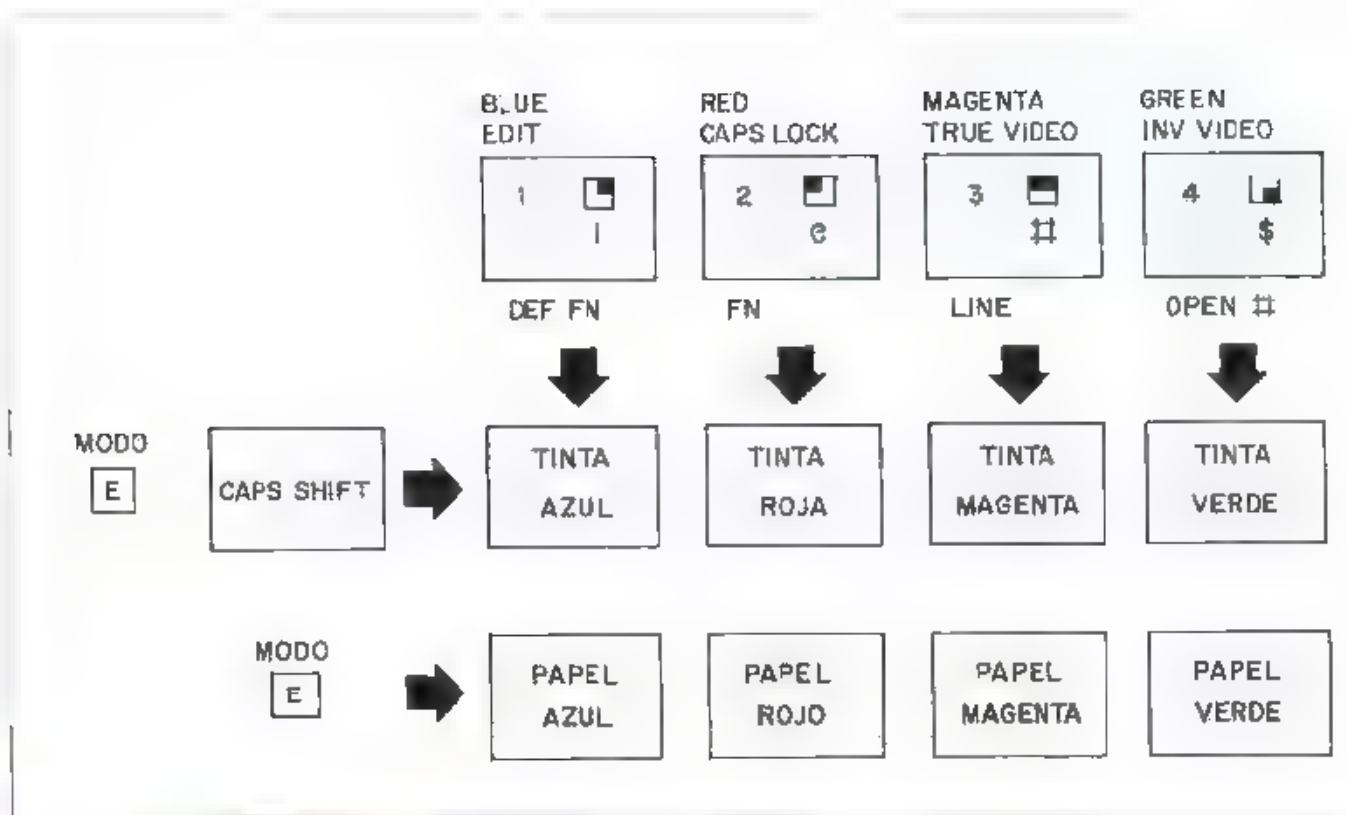
MODO **E**

SYMBOL SHIFT

Definición

El comando «BRILLO» permite modificar el brillo de los colores.

La estructura general de esta sentencia es.



Acceso directo al color

SENTENCIA	ARGUMENTO
BRIGHT	Código de control

Ejemplos:

- BRIGHT 0
 - PRINT BRIGHT 1; "HOLA"
 - BRIGHT 1
 - INPUT BRIGHT 1; ">"; a
- La opción por defecto es brillo desactivado. El brillo afecta tanto al color del papel como al de la tinta.

Ejemplo:

```

10 REM *****
11 REM *****
12 REM *****
13 REM *****
14 REM *****
15 REM *****
16 REM *****
17 REM *****
18 REM *****
19 REM *****
20 REM *****
21 REM *****
22 REM *****
23 REM *****
24 REM *****
25 REM *****
26 REM *****
27 REM *****
28 REM *****
29 REM *****
30 REM *****
31 REM *****
32 REM *****
33 REM *****
34 REM *****
35 REM *****
36 REM *****
37 REM *****
38 REM *****
39 REM *****
40 REM *****
41 REM *****
42 REM *****
43 REM *****
44 REM *****
45 REM *****
46 REM *****
47 REM *****
48 REM *****
49 REM *****
50 REM *****
51 REM *****
52 REM *****
53 REM *****
54 REM *****
55 REM *****
56 REM *****
57 REM *****
58 REM *****
59 REM *****
60 REM *****
61 REM *****
62 REM *****
63 REM *****
64 REM *****
65 REM *****
66 REM *****
67 REM *****
68 REM *****
69 REM *****
70 REM *****
71 REM *****
72 REM *****
73 REM *****
74 REM *****
75 REM *****
76 REM *****
77 REM *****
78 REM *****
79 REM *****
80 REM *****
81 REM *****
82 REM *****
83 REM *****
84 REM *****
85 REM *****
86 REM *****
87 REM *****
88 REM *****
89 REM *****
90 REM *****
91 REM *****
92 REM *****
93 REM *****
94 REM *****
95 REM *****
96 REM *****
97 REM *****
98 REM *****
99 REM *****
100 REM *****

```

«BRIGHT» también puede tener como código de control el número «8», con el significado de transparente.

Ejemplo:

```

10 REM *****
11 REM *****
12 REM *****
13 REM *****
14 REM *****
15 REM *****
16 REM *****
17 REM *****
18 REM *****
19 REM *****
20 REM *****
21 REM *****
22 REM *****
23 REM *****
24 REM *****
25 REM *****
26 REM *****
27 REM *****
28 REM *****
29 REM *****
30 REM *****
31 REM *****
32 REM *****
33 REM *****
34 REM *****
35 REM *****
36 REM *****
37 REM *****
38 REM *****
39 REM *****
40 REM *****
41 REM *****
42 REM *****
43 REM *****
44 REM *****
45 REM *****
46 REM *****
47 REM *****
48 REM *****
49 REM *****
50 REM *****
51 REM *****
52 REM *****
53 REM *****
54 REM *****
55 REM *****
56 REM *****
57 REM *****
58 REM *****
59 REM *****
60 REM *****
61 REM *****
62 REM *****
63 REM *****
64 REM *****
65 REM *****
66 REM *****
67 REM *****
68 REM *****
69 REM *****
70 REM *****
71 REM *****
72 REM *****
73 REM *****
74 REM *****
75 REM *****
76 REM *****
77 REM *****
78 REM *****
79 REM *****
80 REM *****
81 REM *****
82 REM *****
83 REM *****
84 REM *****
85 REM *****
86 REM *****
87 REM *****
88 REM *****
89 REM *****
90 REM *****
91 REM *****
92 REM *****
93 REM *****
94 REM *****
95 REM *****
96 REM *****
97 REM *****
98 REM *****
99 REM *****
100 REM *****

```

INVERSE

Acceso al teclado



MODO E

Definición

Básicamente esta sentencia intercambia los colores de la tinta y papel. Véase en la página 70 la forma de acceder directamente a esta función y en la 81 una figura a u-siva.

Su estructura general es:

SENTENCIA	ARGUMENTO
INVERSE	Código de control

Ejemplos:

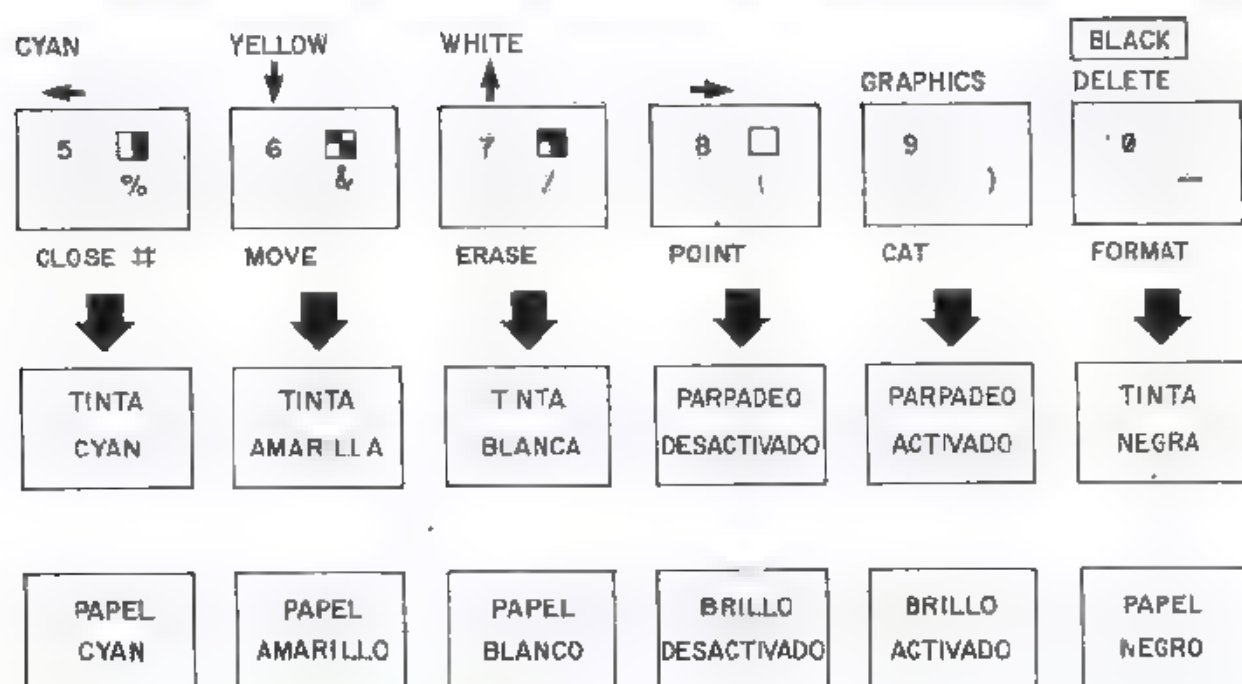
- INVERSE 1
- PRINT INVERSE 1; "JUAN"
- INPUT INVERSE 1; "edad?"; a
- INVERSE 0

También en este caso la opción por defecto es "VIDEO TRUE". Veamos un ejemplo.

```

10 REM *****
11 REM *****
12 REM *****
13 REM *****
14 REM *****
15 REM *****
16 REM *****
17 REM *****
18 REM *****
19 REM *****
20 REM *****
21 REM *****
22 REM *****
23 REM *****
24 REM *****
25 REM *****
26 REM *****
27 REM *****
28 REM *****
29 REM *****
30 REM *****
31 REM *****
32 REM *****
33 REM *****
34 REM *****
35 REM *****
36 REM *****
37 REM *****
38 REM *****
39 REM *****
40 REM *****
41 REM *****
42 REM *****
43 REM *****
44 REM *****
45 REM *****
46 REM *****
47 REM *****
48 REM *****
49 REM *****
50 REM *****
51 REM *****
52 REM *****
53 REM *****
54 REM *****
55 REM *****
56 REM *****
57 REM *****
58 REM *****
59 REM *****
60 REM *****
61 REM *****
62 REM *****
63 REM *****
64 REM *****
65 REM *****
66 REM *****
67 REM *****
68 REM *****
69 REM *****
70 REM *****
71 REM *****
72 REM *****
73 REM *****
74 REM *****
75 REM *****
76 REM *****
77 REM *****
78 REM *****
79 REM *****
80 REM *****
81 REM *****
82 REM *****
83 REM *****
84 REM *****
85 REM *****
86 REM *****
87 REM *****
88 REM *****
89 REM *****
90 REM *****
91 REM *****
92 REM *****
93 REM *****
94 REM *****
95 REM *****
96 REM *****
97 REM *****
98 REM *****
99 REM *****
100 REM *****

```

```
20 BORDER 2 PAPER 2 INK 0 0
RIGHT 0 CLS
40 PRINT AT 10 10 ZX SPECTRUM
40 PRINT AT 14 10 INVERSE 1
ZX SPECTRUM
```

FLASH

Acceso al teclado

L LIST



FLASH

Definición

Con la sentencia «FLASH» se controla la característica

de parpadeo de los caracteres. Secuencialmente se van intercambiando los colores de tinta y papel.

Su estructura general es la siguiente

SENTENCIA	APLICAMENTO
FLASH	código de control

Ejemplos

- FLASH 1
- PRINT FLASH 1; "HOP"
- FLASH 0
- INPUT FLASH 1; "? ", a

El atributo por defecto es parpadeo desactivado. Las sentencias «FLASH» y «BRIGHT» al igual que «PAPER» e «INK» controlan las características de impresión de caracteres completos, es decir, de bloques de 64 pixels (8 × 8).

Introduzca los siguientes comandos directos que harán parpadear a toda la pan-

talla, puesto que se modifican sus atributos

FLASH 1 CLS

Veamos otro ejemplo

```
10 REM *****
   REM PARPADEO
   REM *****

20 BORDER 2 PAPER 2 INK 0 0
40 PRINT AT 10 10 MICROMANIA
40 PRINT AT 14 10 FLASH 1
MICROMANIA
```

«FLASH» también tiene la posibilidad de conservar los atributos de pantalla con el código de control «8» (transparencia)

16	TINTA
17	PAPEL
18	PARPADEO
19	PRINTEO
20	INVERTIDO



Posibilidades de color en un juego.

Para introducir estos códigos es necesario utilizar la función «CHR\$».

Ejemplo.

```
PRINT CHR$ 16; CHR$ 5; "PEPE"
```

El primer carácter de control corresponde al de tinta y el segundo al código de color, la cadena «PEPE» será visualizada con tinta amarilla (5). Consiguiendo el mismo resultado que

```
PRINT INK 5; "PEPE"
```

El uso de estos caracteres puede tener utilidad en la asignación de atributos a variables de cadena

Ejemplo.

```
LET a$ = CHR$ 16 + CHR$ 0 +  
"MICRO"  
LET b$ = CHR$ 16 + CHR$ 2 +  
"HOBBY"  
LET c$ = a$ + b$  
PRINT c$
```

Una parte de la variable «c\$» se imprime en tinta negra y la otra en roja.

Se podría haber asignado directamente la totalidad de los caracteres de control a la variable c\$.

Acceso directo

Sin hacer uso de las sentencias «PAPER», «INK»,

«FLASH» o «BRIGHT» se pueden utilizar de forma directa, los colores

Pase a modo **E** (extendido) y pulse, por ejemplo, la tecla «1»; ¿qué ocurre?, simplemente que a partir de ese instante escribe con fondo azul, y si vuelve al modo **E** y pulsa simultáneamente «CAPS SHIFT» y la tecla «7», la tinta será blanca.

En el modo extendido se tiene un acceso directo, con las teclas de la fila superior, a todas las combinaciones de color de papel y tinta y a la activación o desactivación de las características de parpadeo y brillo

En la figura adjunta se puede identificar la forma de ac-

ceder a cualquiera de estas combinaciones

Entre las aplicaciones de acceso directo, puede destacar la utilidad que tiene el editar un programa con diversos colores, para resaltar u ocultar alguna de sus zonas

El acceso directo a las funciones de VIDEO fue revisado en la página 70

Errores

Si se especifica un argumento erróneo en alguna de estas sentencias, se producen los siguientes errores

a) Color no válido

K Invalid colour

Ejemplos.

```
BORDER 8
PAPER 10
INVERSE 2
FLASH 8
```

Ejemplo programa 4.

b) Entero fuera de rango

R Integer range

Ocorre cuando el código de color es inferior a «0» o superior a «255».

Ejemplos:

```
INK -1
BRGHT 200
```

Una instrucción del tipo «BORDER 3.6» no es errónea ya que el argumento queda redondeado a «4», obteniéndose por tanto un borde de color verde ■

PROGRAMA 4

```
10 REM *****
  * ATRIBUTOS *
  * ***** *
20 BORDER 4 PAPER 4 INK 0 C
LS
25 REM *****
30 INPUT "ATRIBUTO ", LINE a
$
40 IF a$="" THEN GO TO 30
50 FOR n=1 TO LEN a$
60 IF a$(n) < "0" OR a$(n) > "9" T
HEN GO TO 30
70 NEXT n
80 LET atributo=VAL a$
90 IF atributo<0 OR atributo>2
55 THEN GO TO 30
100 PRINT AT 1,0, "CODIGO DECIMA
L ... ";atributo
102 REM *****
104 LET numero=atributo
106 LET b$=""
110 DIM i(8)
120 FOR n=7 TO 0 STEP -1
130 IF numero=INT (2^n) THEN L
ET numero=numero-INT (2^n) LET
i(n+1)=1
140 LET b$=b$+STR$ (i(n+1))
150 NEXT n
160 PRINT "CODIGO BINARIO ...
;b$
170 REM *****
180 IF b$(1)=1 THEN PRINT FLA
SH 1, "PARPADEO ACTIVADO" GO
TO 200
```

```
190 PRINT "PARPADEO DESACTIVADO"
200 IF b$(2)="1" THEN PRINT BR
GHT 1, "BRILLO ACTIVADO" GO T
O 212
210 PRINT "BRILLO DESACTIVADO"
212 LET papel=0
214 LET peso=0
220 FOR n=5 TO 3 STEP -1
230 LET valor=VAL b$(n)
240 IF valor=1 THEN LET papel=p
apel+2^peso
244 LET peso=peso+1
250 NEXT n
260 PRINT "PAPEL ";papel; INK 9, "P
APER ... ";papel
270 LET tinta=0
280 LET peso=0
290 FOR n=6 TO 4 STEP -1
300 LET valor=VAL b$(n)
310 IF valor=1 THEN LET tinta=t
inta+2^peso
320 LET peso=peso+1
330 NEXT n
340 PRINT "PAPER ";papel; INK tinta, 'T
INTA ... ";tinta
350 REM *****
360 PRINT #0, "Otro atributo (5/
000)";
364 LET z$=INKEY$
370 IF z$="n" OR z$="N" THEN ST
OP
380 IF z$="s" OR z$="S" THEN GO
TO 20
390 GO TO 362
```


PROGRAMA DEPURACION

```

9979 STOP
9980 REM DEPURACION BASIC
9983 LET err SP=(PEEK 23614) *256
+ (PEEK 23613)
9984 POKE 23692,1
9985 LET Linea=(PEEK (err sp+2))
+256+(PEEK (err sp+2))
9986 LET posicion=PEEK (err sp+4)
+2
9987 OPEN #2,"K"
9988 PRINT AT 0,0,"Parada en ",Linea,
posicion
9989 PAUSE 0
9990 IF INKEY$="V" OR INKEY$="U"
THEN GO SUB 9993

```

```

9991 CLOSE #2
9992 INPUT 0 RETURN
9993 INPUT AT 0,0,"Variable: ",
LINE V$
9994 IF V$="" THEN INPUT 0 RETURN
9995 PRINT AT 0,0,"Contenido: ";V$

```

```

9996 IF LEN V$=2 THEN IF V$(2)=""
THEN PRINT "VAL: VAL$ V$,"
GO TO 9998
9997 PRINT VAL V$
9998 PAUSE 0
9999 GO TO 9993

```

Capitulo Depuración programas

PROGRAMA 5

```

10 REM *****
* CURSO/BASIC *
* *****
* CARTA COLOR *
* *****
20 BORDER 4 PAPER 4 INK 0 C
30 RESTORE
40 FOR X=0 TO 31 STEP 2
50 READ color
60 FOR Y=0 TO 21
70 PRINT PAPER color,AT Y,X,

```

```

80 NEXT Y
90 NEXT X
95 REM CONTORNO
100 PLOT 0,0
110 DRAW 0,175
120 DRAW 255,0
130 DRAW 0,175
140 DRAW 255,0
145 PRINT #0,"Sintoniza correc
tamente la T"
150 PAUSE 0
160 REM TABLA COLORES
170 DATA 0,1,2,3,4,5,6,7,0,1,2,
3,4,5,6,7

```

PROGRAMA 6

```

10 REM *****
* CURSO/BASIC *
* *****
* COLORES 1 *

```

```

* *****
20 FOR n=0 TO 6 STEP 2
30 POKE USR "a"+n,85
40 POKE USR "a"+n+1,170
50 NEXT n
60 NEXT

```

PROGRAMA 7

```

100 REM *****
* CURSO/BASIC *
* *****
* COLORES 2 *
* *****
102 BORDER 4 PAPER #: INK 0 C
103 GO SUB 1000
104 LET a$=CHR$ 144
105 LET brillo=0
110 FOR y=3 TO 17 STEP 2
120 FOR x=3 TO 22 STEP 2
130 PRINT BRIGHT brillo, PAPER
(y-3)/2, INK (x-3)/2,AT y,x,a$,A
(y,x+1,a$,AT y+1,x,a$,AT y+1,x+
1,a$
140 NEXT x
150 NEXT y
160 PRINT #0,"F" Código /
ENTER Continúa
170 PAUSE 0
180 INPUT 0

```

```

182 IF INKEY$="F" THEN GO SUB 1
200
190 IF brillo=0 THEN LET brillo
=1 PRINT AT 0,0,"BRILLO" GO TO
110
200 PRINT AT 0,0,"" GO T
O 106
1000 REM NUMEROS
1010 PRINT INVERSE 1,AT 0,0,"
TINTA
1020 FOR x=9 TO 23 STEP 2
1040 PRINT AT 2,x,(x-9)/2
1060 NEXT x
1064 LET b$=" PAPEL
1065 FOR y=3 TO 18
1080 PRINT INVERSE 1,AT y,5,b$(y
-2)
1072 NEXT y
1090 FOR y=4 TO 18 STEP 2
1100 PRINT AT y,7,(y-4)/2
1120 NEXT y
1130 RETURN
1200 REM PANTALLA
1210 INPUT "Codigo >>> "; LINE c$
1212 IF LEN c$<>2 THEN GO TO 121

```

```

0
1220 IF VAL C$(1) < 0 OR VAL C$(1)
> 7 THEN GO TO 1210
1230 LET papel=VAL C$(1)
1240 IF VAL C$(2) < 0 OR VAL C$(2)
> 7 THEN GO TO 1210
1250 LET tinta=VAL C$(2)
1255 LET d$=
1264 FOR n=1 TO 32
1266 LET d$=d$+a$
1268 NEXT n

```

```

1260 FOR n=0 TO 21
1270 PRINT PAPER papel, INK tinta
: BRIGHT brillo, AT n, 0 d$
1280 NEXT n
1290 PRINT #0, "Pulse una tecla
para continuar."
1300 PAUSE 0
1310 CLS
1312 GO SUB 1000
1320 RETURN

```

GRAFICOS

La posibilidad de realizar gráficos con un ordenador como el Spectrum, es muy valioso para el usuario ya que entre las aplicaciones de éstos, unas veces como parte importante y otras como complemento a los programas, merecen destacarse:

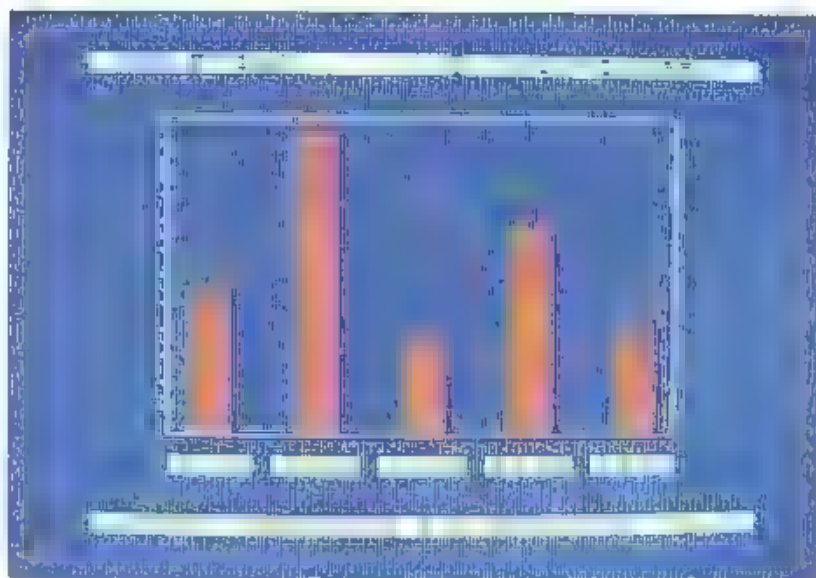
- Posibilidades gráficas en juegos
- Presentación de gráficos de gestión (histogramas, gráficos de tarta, etc.).
- Utilización en el diseño industrial, esta aplicación es conocida por las siglas CAD-CAM, Computer Aided Design y Computer Aided Manufacturing, que traducido al español viene a significar: Diseño asistido por ordenador y fabricación asistida por ordenador.
- Matemáticas (Dibujo de Funciones).

Tipos de Gráficos

Atendiendo a la resolución gráfica de los dibujos, éstos se pueden clasificar en

Gráficos de baja resolución
Gráficos de alta resolución

Para la realización de dibujos en baja resolución nue-



Gráficos de gestión.

den utilizarse bloques de color o los gráficos predefinidos; en cambio para los gráficos de alta resolución el Spectrum dispone de las sentencias:

PLOT
DRAW
GRAPH

Estas sentencias pueden ser combinadas con «PAPER», «INK», «FLASH», «INVERSE» y «OVER».

Bloques de color

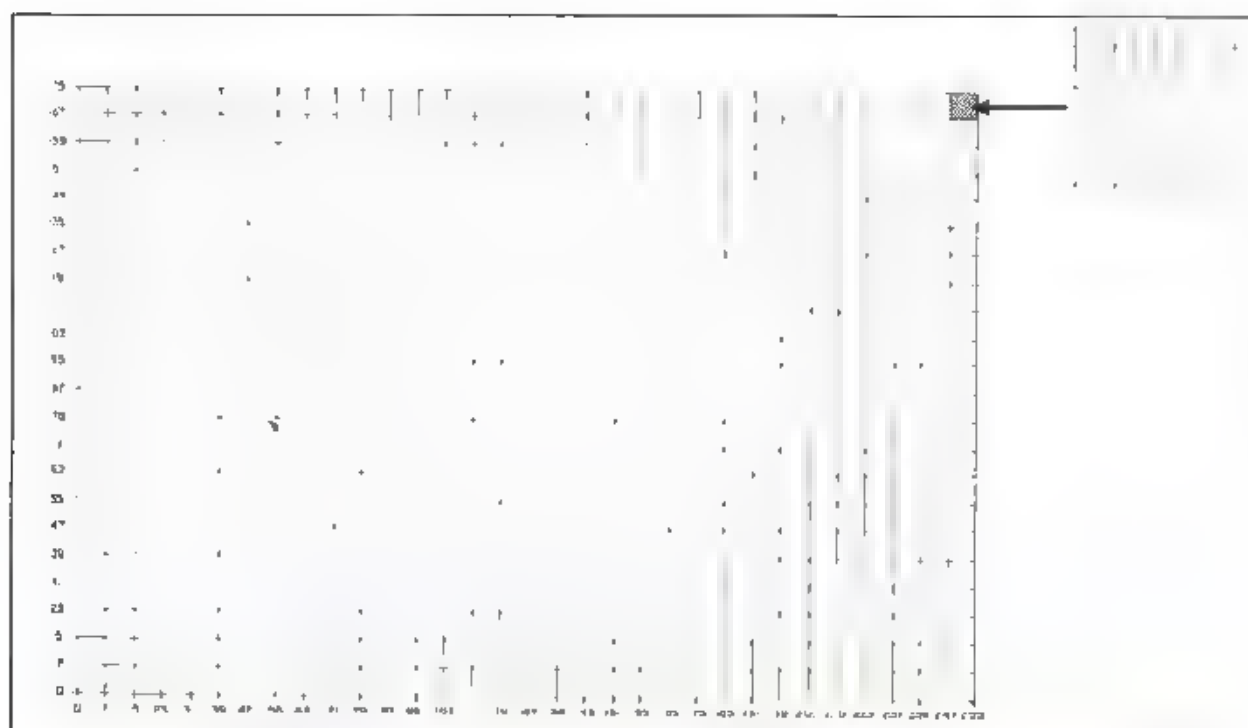
Utilizando el carácter «espacio», con diversos colo-

res de fondo (PAPER), se pueden conseguir gráficos a base de bloques coloreados de 8 por 8 pixel.

El programa número «1» dibuja en pantalla un simpático pez. El número «2» visualiza el nombre de la revista así como las bandas coloreadas características de Spectrum

Gráficos predefinidos

Para realizar nuestros dibujos en baja resolución, también podemos utilizar los gráficos que se encuentran predefinidos en el juego de caracteres del Spectrum. Pa-



Pantalla en alta resolución.

PROGRAMA 1

```

10 REM *****
  * CURSO/BASIC *
  * ***** *
  * DIBUJO PEZ *
  * ***** *
20 BORDER 5 PAPER 3 INK 0 C
LS
30 REM *****
  * PRINCIPAL *
  * ***** *
32 RESTORE
40 LET color=4
50 GO SUB 1000
60 LET color=5
70 GO SUB 1000
80 LET color=3
90 GO SUB 1000
100 LET color=7
110 GO SUB 1000
120 LET color=0
130 GO SUB 1000
140 LET color=2
150 GO SUB 1000
160 PAUSE 100
170 REM *****
  * BURBUJAS *
  * ***** *
180 FOR n=1 TO 9
190 READ y,x
200 PRINT PAPER 1,AT y,x,"
210 PAUSE 25
220 NEXT n
230 PAUSE 0
240 STOP
250 REM

```

```

*****
* SUB. DIBUJO *
*****
1010 READ y,x
1020 IF NOT y THEN RETURN
1030 PRINT PAPER color;AT y,x;"
1040 GO TO 1010
2000 REM *****
  * TABLA DE DATOS *
  * ***** *
2010 DATA 5,19,5,20,5,21,6,16,6,
17,6,16,6,19,6,20,6,21,7,13,7,14,
7,15,7,16,7,17,7,18,7,19,7,20,
2020 DATA 8,12,8,13,8,14,8,15,8,
16,8,17,8,18,8,19,8,20,9,11,9,12,
9,13,9,14,9,15,9,16,9,17,9,18,9,
19,9,20,
2030 DATA 10,12,10,13,10,14,10,1
5,10,16,10,17,10,18,10,19,10,20,
11,13,11,14,11,15,11,16,11,17,11,
18,11,19,11,20,
2040 DATA 12,12,12,13,12,14,12,1
5,12,16,12,17,12,18,12,19,12,20,
13,11,13,12,13,13,13,14,13,15,13,
16,13,17,13,18,13,19,13,20,
2050 DATA 14,13,14,14,14,15,14,1
6,14,17,14,18,14,19,14,20,15,14,
15,15,15,16,15,17,15,18,15,19,15,
20,
2060 DATA 16,14,16,15,16,16,16,1
7,16,18,16,19,16,20,16,21,17,16,
17,17,17,18,17,19,17,20,17,21,
2070 DATA 18,19,18,20,18,21,13,2
3,13,24,14,24,14,25,15,24,15,25,
16,24,0,0,
2080 DATA 5,22,6,22,6,23,6,24,7,
21,7,22,7,23,7,24,8,21,8,22,8,23,
8,24,8,25,9,21,9,25,10,21,10,25,
11,21,11,25,
2090 DATA 12,21,13,21,13,22,14,2

```

```


1,14,22,14,23,15,21,15,22,15,23,
16,22,15,23,17,22,17,23,0,0
2100 DATA 11,26,12,26,12,26,12,2
7,13,25,14,26,14,27,11,12,11,11,
11,10,11,9,10,11,10,10,9,10,8
2110 DATA 12,11,12,10,12,9,12,8,
12,7,9,9,9,8,9,7,9,6,13,9,13,8,1
3,7,13,6,13,5,8,9,8,8,8,7,8,6,8,
5,8,4
2120 DATA 14,7,14,6,14,5,14,4,14
,3,7,4,7,3,7,2,15,5,15,4,15,3,15
,2,15,1,0,0
2130 DATA 9,22,9,23,9,24,10,22,1

```

```

0,24,11,22,11,24,12,22,12,23,12,
24,0,0
2140 DATA 10,23,11,23,0,0
2150 DATA 2,19,3,18,3,19,3,20,4,
17,4,18,4,19,4,20,4,21,5,15,5,16
,5,17,5,18,6,14,6,15
2160 DATA 17,15,18,9,18,10,18,11
,18,12,19,13,18,14,18,15,18,16,1
8,17,18,18
2170 DATA 19,14,19,15,19,16,19,1
7,20,13,20,14,20,15,20,16,0,0
2180 DATA 13,29,12,31,10,30,9,26
,7,30,5,29,3,28,2,30,0,31

```

ra acceder a ellos debe seleccionarse el modo «gráfico» (); en la página 7 de este manual se explica con detalle dicho procedimiento.

El programa número «3» genera una serie de dibujos aleatorios utilizando estos gráficos, el color del «papel» es también aleatorio y la «tinta» tiene atributo de contraste. La visualización de estos gráficos se realiza con la función «CHR\$».

Dentro del código ASCII del Spectrum, los gráficos predefinidos tienen un código comprendido entre el 128 y el 143 en decimal.

El programa número «4», utilizando los gráficos predefinidos, visualiza un dibujo infantil de nuestra redacción

Pantalla en alta resolución

Cuando se utiliza la pantalla en la modalidad de alta resolución, el eje de abscisas (x) se divide en 256 píxel y el de ordenadas (y) en 176, esto nos da un total de 45056 píxel.

El origen del eje de coordenadas de la pantalla en alta resolución se encuentra en el ángulo inferior izquierdo, a diferencia del de baja resolución que se encuentra en el ángulo superior izquierdo.

PLOT

Acceso al teclado

SIN



ASN

Tipo de sentencia

Comando de dibujo

Definición

La sentencia «PLOT» visualiza un píxel, determinado por sus coordenadas «x» e «y», del color especificado de «tinta».

La estructura general de la sentencia es:

SENTENCIA	ARGUMENTO
PLOT	COORD. X, COORD. Y

Ejemplos.

- PLOT 100, 100
- PLOT PAPER 4, 20, 30
- PLOT INK 5; 127, 30
- PLOT a, b

El siguiente programa genera una serie de puntos aleatorios, de distinto color, en la pantalla

```

10 REM *****
   : PUNTOS :
   : *****
20 BORDER 5 PAPER 5, CLS
30 LET X=INT (RND*256)
40 LET Y=INT (RND*176)
50 LET COLOR=INT (RND*63)
60 PLOT INK COLOR X,Y
70 GO TO 30

```

Observará, cuando la pantalla tiene cierta cantidad de puntos, que al visualizar uno nuevo cambian de color los de alrededor, esto es debido a que la representación del color se hace en baja resolución, por tanto todos los píxel (64) de un bloque de caracteres deben tener el mismo color.

Con los siguientes programas se pueden dibujar punto a punto las gráficas de las funciones «SENO» y «COSENO»; para que pueda ser visualizado un ciclo completo (360°) ha sido necesario calcular los «puntos» cada dos grados.

a) Función «SENO»:

```

10 REM *****
   : FUNCION SENO :
   : *****
20 DEF F=X*(PI/180)
30 FOR N=0 TO 180
40 LET Y=FN F*10
50 PLOT N, Y*60+40
60 NEXT N

```

b) Función «COSENO»


```

10 REM *****
11 FUNCTION CODENO
12 *****
13 LET origen=0
14 LET amplitud=50
15 DEF FN A(X)=COS X*PI/180
16 FOR n=0 TO 100
17 LET y=FN A(n*2)
18 PLOT n,origen+ampli*tody
19 NEXT n

```

DRAW

Acceso al teclado

COS



ACS

Tipo de sentencia

Comando de dibujo

Definición

Con «DRAW» se pueden dibujar líneas rectas y curvas (arcos). Su estructura general es

SENTENCIA	ARGUMENTO
DRAW	coord. x, coord y z

El parametro «Z» es opcional y sirve para dibujar arcos.

Ejemplos:

DRAW 40, 30

DRAW INK 8; -20, 10

- DRAW 10, 50, 7

- DRAW 7, 10, -3

El punto de origen de una línea es el ultimo pixel visualizado, bien sea por una sentencia «PLOT», «DRAW» o

Plot y Draw.

«CIRCLE» y el punto de destino es el especificado por las coordenadas (relativas al punto de origen) del argumento de «DRAW», por ejemplo las sentencias

```

PLOT ,00
DRAW b0 0

```

visualiza una recta entre los

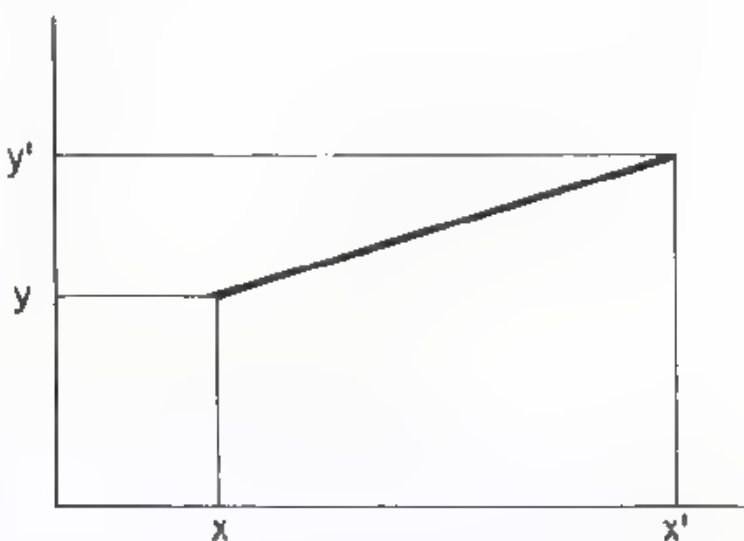
puntos cuyas coordenadas son: a (100, 100) y b (160, 90)

Las sentencias «RUN», «CLS», «NEW» y «CLEAR» posicionan el origen, por defecto, en las coordenadas 0, 0.

El programa número «5» visualiza una serie de rectas aleatorias de diversos colores; pulsando la tecla «P» (pa-



PLOT x,y



PLOT x,y : DRAW x',y'

PROGRAMA 2

```

10 REM *****
  * CURSOR/BASIC *
  * ***** *
  * MICROHOBBY *
  * *****
15
12 BORDER 1: PAPER 1: INK 0: C
14 REM
  * *****
  * BORDE *
  * *****

20 FOR n=0 TO 31
30 PRINT AT 0,n, PAPER 0, ' '
40 PRINT AT 21,n, PAPER 0, ' '

50 NEXT n
60 FOR n=1 TO 20
70 PRINT AT n,0, PAPER 0, ' '
80 PRINT AT n,31, PAPER 0, ' '
90 NEXT n
100 REM
  * *****
  * MICRO *
  * *****

110 RESTORE
120 READ y,x
130 IF NOT y THEN GO TO 240
140 PRINT AT y,x, PAPER 0, " "
150 GO TO 120
150 REM
  * *****
  * DATOS *
  * *****

170 DATA 7,5,6,5,5,5,5,6,5,7,6,
7,7,7,6,8,5,9,6,9,7,9
180 DATA 9,11,6,11,7,11,7,12,7
13,3,11
190 DATA 5,17,5,16,5,15,6,15,7
15,7,16,7,17
200 DATA 7,19,6,18,6,19,6,20,5,
21
210 DATA 7,23,6,23,5,23,5,24,5,
25,6,25,7,25,7,24
220 DATA 0,0
230 REM
  * *****
  * HOBBY *
  * *****

```

```

250 READ y,x
260 IF NOT y THEN GO TO 350
270 PRINT PAPER 2, AT y,x, "
280 GO TO 250
290 REM
  * *****
  * DATOS *
  * *****

```

```

300 DATA 15,5,14,6,13,6,12,6,11
6,13,7,13,8,14,8,15,8
310 DATA 15,10,14,10,13,10,13,1
1,13,12,14,12,15,12,15,11
320 DATA 11,14,12,14,13,14,14,1
4,15,14,15,16,15,16,14,16,13,16,
13,15
330 DATA 11,18,12,18,13,18,14,1
5,15,16,15,19,13,20,14,20,13,20,
13,19
340 DATA 13,22,14,22,15,22,15,2
3,15,24,14,24,13,24,16,24,17,24
17,23,17,22

```

```

350 DATA 0,0
360 REM
  * *****
  * ESPECTRO *
  * *****

```

```

370 LET posicion=24
380 LET color=2
390 GO SUB 1000
400 LET color=6
410 GO SUB 1000
420 LET color=4
430 GO SUB 1000
440 LET color=5
450 GO SUB 1000
460 PAUSE 0
470 CLS
480 STOP
490 REM
500 REM
  * *****
  * SUBROUTINA *
  * *****

```

```

1010 LET y=20
1020 FOR x=posicion TO 30
1030 PRINT PAPER color, AT y,x,
"
1040 LET y=y-1
1050 NEXT x
1055 LET posicion=posicion+1
1060 RETURN

```

rar) dejan de generarse, y con la opción «C» continúa.

Para comprobar la resolución gráfica de las rectas, edite el siguiente programa que «rota» una recta sobre un punto, con un incremento de cinco grados.

```

10 REM *****
  * ROTACION *
  * *****
15
20 BORDER 4: PAPER 4: INK 0: C
30 LET A=INT(PI)=3.14159

```

```

40 LET y=INT(21)=20
50 LET longitud=80
60 FOR n=0 TO 360 STEP 5
70 LET rad=PI/180
80 LET x=COS rad*longitud
90 LET y=SIN rad*longitud
100 PLOT x,y
110 DRAW x,y
120 NEXT n

```

Observará que la resolución depende de la inclinación de la recta sobre la horizontal.

El siguiente programa genera una serie de cuadrados crecientes a partir de la esquina inferior izquierda.

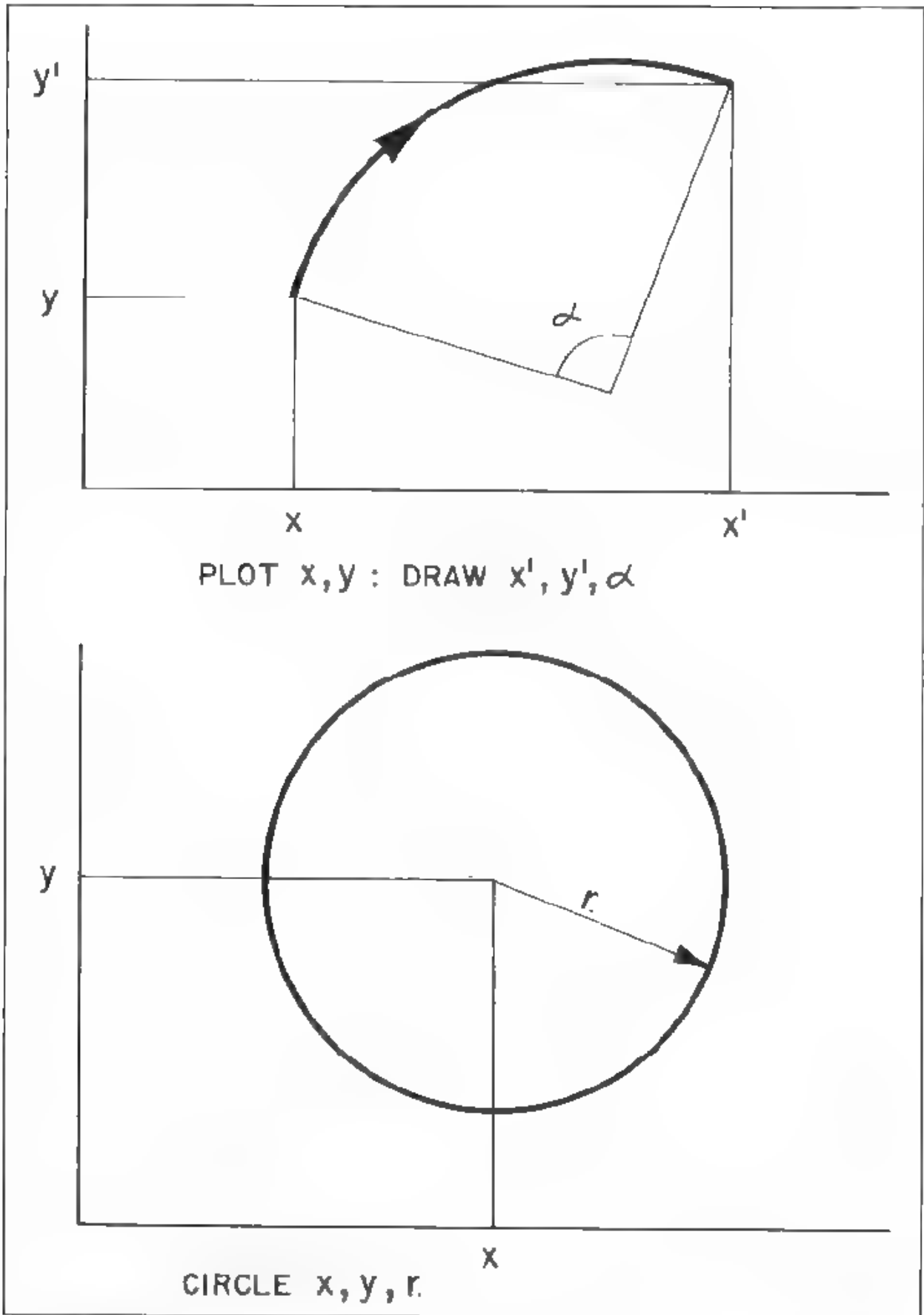
```

10 REM *****
  * CUADRADOS *
  * *****
15
20 BORDER 5: PAPER 5: INK 0: C
30 INPUT "Incremento >>>"; paso
40 FOR n=1 TO 80 STEP paso
50 PLOT 0,n
60 DRAW 0,n
70 DRAW n,0
80 DRAW n,n
90 NEXT n

```

La variable «paso» indica la diferencia entre dos cuadrados consecutivos.

Este otro genera unos rectángulos concéntricos.



Draw y Circle.

```

10 REM *****
11 RECTANGULO$ = " "
12 *****
13
14 BORDER 5 PAPER 8 INK 0 C
15
16 INPUT "Incremento >>> ", pas
17
18 LET base=235
19 LET altura=155
20 FOR n=10 TO 80 STEP pas
21   PLOT n
22   DRAW base, 0
23   DRAW 0, altura
24   DRAW -base, 0
25   DRAW 0, -altura
26   LET base=base+pas
27   LET altura=altura-pas
28 NEXT n

```

La variable «paso» tiene el mismo significado que en el programa anterior.

Arcos de circunferencia

Como se explicó anteriormente, el parámetro «Z» de una sentencia «DRAW» era opcional y permitía dibujar arcos de circunferencia (líneas curvas). Este parámetro indica el ángulo de giro expresado en radianes.

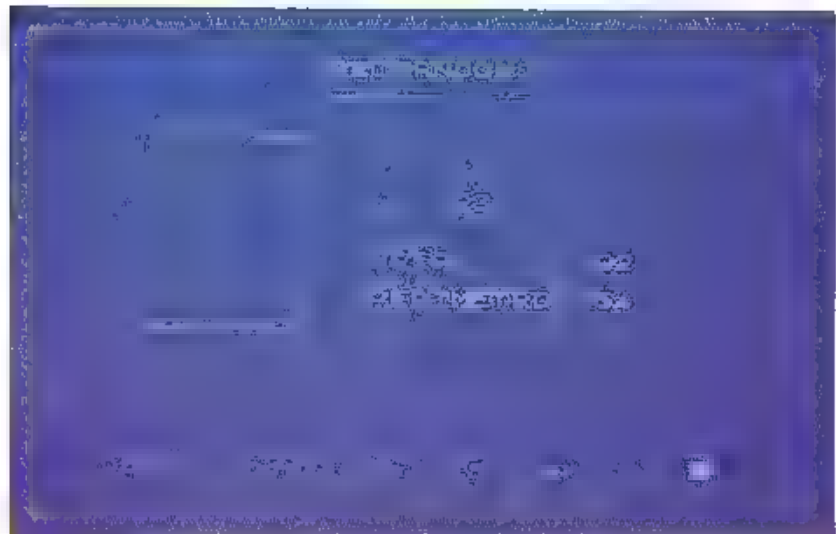
Cuando «Z» es positivo, el giro se realiza en sentido contrario a las agujas del reloj (hacia la izquierda)

Ejemplo

```

PLOT 30 30
DRAW 100, 100 3

```



Aplicaciones en matemáticas.



Gráfico de «baja resolución».

Cuando «Z» es negativo, el giro se realiza en el sentido de

las agujas del reloj (hacia la derecha)

PROGRAMA 3

```

10 REM *****
11 * CURSO/BASIC *
12 * ABSTRACTO *
13 *****
14
15 BORDER 5 PAPER 5 INK 0 C
16
17 RANDOMIZE
18 FOR y=0 TO 31
19   FOR x=0 TO 31
20     LET caracter=INT (RND*2)

```

```

21   LET grafico=INT (RND*15)+12
22   LET color=INT (RND*8)
23   IF caracter=1 THEN PRINT AT
24     y,x, PAPER color,CHR$ grafico
25   GO TO 20
26
27 PRINT AT y,x, '
28 NEXT x
29 NEXT y
30 PRINT #0, " Pulsa una tecla
31 para continuar"
32 PAUSE 0
33 INPUT 0
34 GO TO 20

```


Ejemplo:

```

PLOT 50, 50
DRAW 100, 100, -1

```

El siguiente programa visualiza una esfera con sus husos.

```

10 REM *****
11      : ARCS *****
12
13 BORDER 3 PAPER 3 INK 0 C
14
15 FOR h=0 TO 3.14 STEP 0.2
16   PLOT 120, 0
17   DRAW 0, 175, h
18   PLOT 120, 0
19   DRAW 0, 175, h
20 NEXT h

```

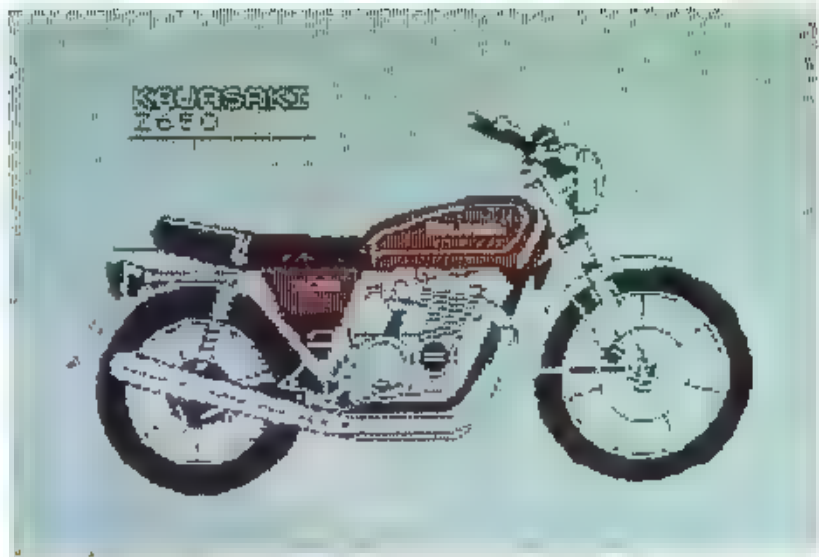


Gráfico de «alta resolución».

Utilizando como parámetro «Z» el valor PI que equivale a 180° una semicircunferencia, el siguiente programa genera una serie de «ondas»:

```

10 REM *****
11      : ONDS *****
12

```

```

20 BORDER 4 PAPER 4, INK 0 C
21
22 FOR x=100 TO 20 STEP -20
23   PLOT 0, 0
24   FOR n=1 TO 5
25     DRAW 20, 0, PI
26   NEXT n
27 NEXT x

```

```

10 REM *****
11      : CURVAS *****
12
13 BORDER 5 PAPER 5 INK 0 C
14
15 INPUT "Angulo de giro: "; angulo
16 CLS
17 PLOT 127, 0
18 DRAW 50, 50, angulo
19 GO TO 20

```

Como efecto curioso de la sentencia «DRAW» ejecute el siguiente programa:

Introduzca, por ejemplo, como datos de «ángulo de gi-

PROGRAMA 4

```

10 REM *****
11      : CURSO/BASIC *****
12      : GRAFICO *****
13
14 BORDER 5 PAPER 5 CLS
15 REM FUNDAMENTO
16 FOR n=10 TO 21
17   PRINT PAPER 2, INK 6, AT n, 0
18 NEXT n
19 REM PARED
20 INK 4
21 PRINT AT 10, 0, "PARED";
22 PRINT "I";
23 PRINT "I";
24 PRINT AT 17, 4, "F";
25 PRINT "I";
26 PAPER 3, INK 1
27 REM EDIFICIO
28 FOR n=17 TO 18
29   PRINT AT n, 10, "EDIFICIO";
30 NEXT n
31 FOR n=15 TO 16
32   PRINT AT n, 10, "EDIFICIO";
33 NEXT n
34 FOR n=8 TO 14
35   PRINT AT n, 10, "F";
36 PRINT "I";
37 PRINT "I";

```

```

38 PRINT "I";
39 PRINT AT 7, 10, "F";
40 PRINT "I";
41 PRINT "I";
42 PRINT AT 7, 10, "F";
43 PRINT "I";
44 INK 2
45 INK 2
46 REM CARTEL
47 PRINT AT 6, 11, PAPER 5, "I";
48 AT 6, 16, "I";
49 PAPER 7
50 PRINT AT 5, 0, "L";
51 PRINT "I";
52 PRINT "I";
53 PRINT AT 4, 0, "I"; AT 4, 19, "I";
54 PRINT AT 3, 0, "I";
55 PRINT AT 4, 9, INK 0, "MICRO";
56
57 PRINT INK 2, "HOBBY";
58 REM COCHE
59 PAPER 5 INK 3
60 PRINT AT 18, 19, "I";
61 PRINT "I";
62 PRINT "I";
63 PRINT AT 17, 20, "I";
64 REM ACERO
65 PRINT AT 18, 28, INK 6, "I";
66 PRINT AT 17, 28, INK 6, "I";
67 INK 4
68 PRINT AT 10, 20, "I";
69 PRINT AT 15, 28, "I";
70 PRINT "I";
71 PAUSE 0

```

ro» los valores

60. 250. 300. 400. 500. 600.
700. 800. y 803

Algunos valores pueden dar el mensaje de error:

B Integer out of range

¡¡¡Sorprendente, verdad!!!

Programa especial

El programa número «6» dibuja el mapa de España punto a punto; para su confección se ha utilizado principalmente la sentencia «PLOT».

El número total de puntos es

Península	718
Baleares	93
Canarias	158
Rótulo	140
Total	1109

Una vez visualizado el dibujo, podrá almacenar el contenido de la pantalla de la forma:

Resolución Gráfica

SAVE «mapa» SCREEN\$

Así podrá utilizarlo en cualquier programa «educativo» que se le ocurra.

CIRCLE

Acceso al teclado

SQR

MODO **E**
SYMBOL
SHIFT



CIRCLE

PROGRAMA 5

```
10 REM *****
  *          *
  *  CURSO/BASIC  *
  *          *
  * *****      *
  *          *
  *    RECTAS      *
  *          *
  * *****      *
20 BORDER 1: PAPER 5: CLS
22 LET x origen=0
24 LET y origen=0
26 RANDOMIZE
30 GO SUB 1000
32 LET x origen=x aleatorio
34 LET y origen=y aleatorio
40 PLOT x aleatorio,y aleatori
42 PRINT #0;AT 1,3;"P = Para
  C = Continúa!
50 GO SUB 1000
60 GO SUB 1100
62 IF color=5 THEN GO TO 60
70 LET x final= x aleatorio-x
```

```
origen
80 LET y final=y aleatorio-y
origen
90 LET x origen=x aleatorio
100 LET y origen=y aleatorio
110 DRAW INK color,x final,y final
112 IF INKEY$="p" OR INKEY$="P"
  THEN GO SUB 1200
120 GO TO 50
999 STOP
1000 REM SUBROUTINA COMPENDIAR
1010 LET x aleatorio=INT (RND*25
6)
1020 LET y aleatorio=INT (RND*17
5)
1030 RETURN
1100 REM SUBROUTINA COLOR
1110 LET color=INT (RND*8)
1120 RETURN
1200 REM SUBROUTINA PAPER
1210 IF INKEY$="c" OR INKEY$="C"
  THEN RETURN
1220 GO TO 1210
```

Tipo de sentencia

Comando de dibujo

Definición

La sentencia «CIRCLE» dibuja un círculo. Su estructura es

SENTENCIA	ARGUMENTO
CIRCLE	coord. x, coord. y radio

Las coordenadas «x» e «y» corresponden a su centro.

Ejemplos:

- CIRCLE 128, 88, 30
- CIRCLE FLASH 1; 100, 100, 50
- CIRCLE a, b, c
- CIRCLE INK 3, 100, 100, 30

El siguiente programa presenta la diferencia entre un círculo dibujado a base de puntos y el visualizado con «CIRCLE».

```

10 REM *****
   :          : CIRCULO :
   :          :          :
12 BORDER 4 PAPER 4 INK 0 C
L3
20 LET x=initial=60
30 LET y=initial=80
40 LET radio=50
50 FOR j=1 TO 360
60 LET radian=PI/180
70 LET x=COS radian*radio
80 LET y=SIN radian*radio
90 PLOT (x initial+x), (y initial+y)
100 NEXT j
110 CIRCLE 100,80,50

```

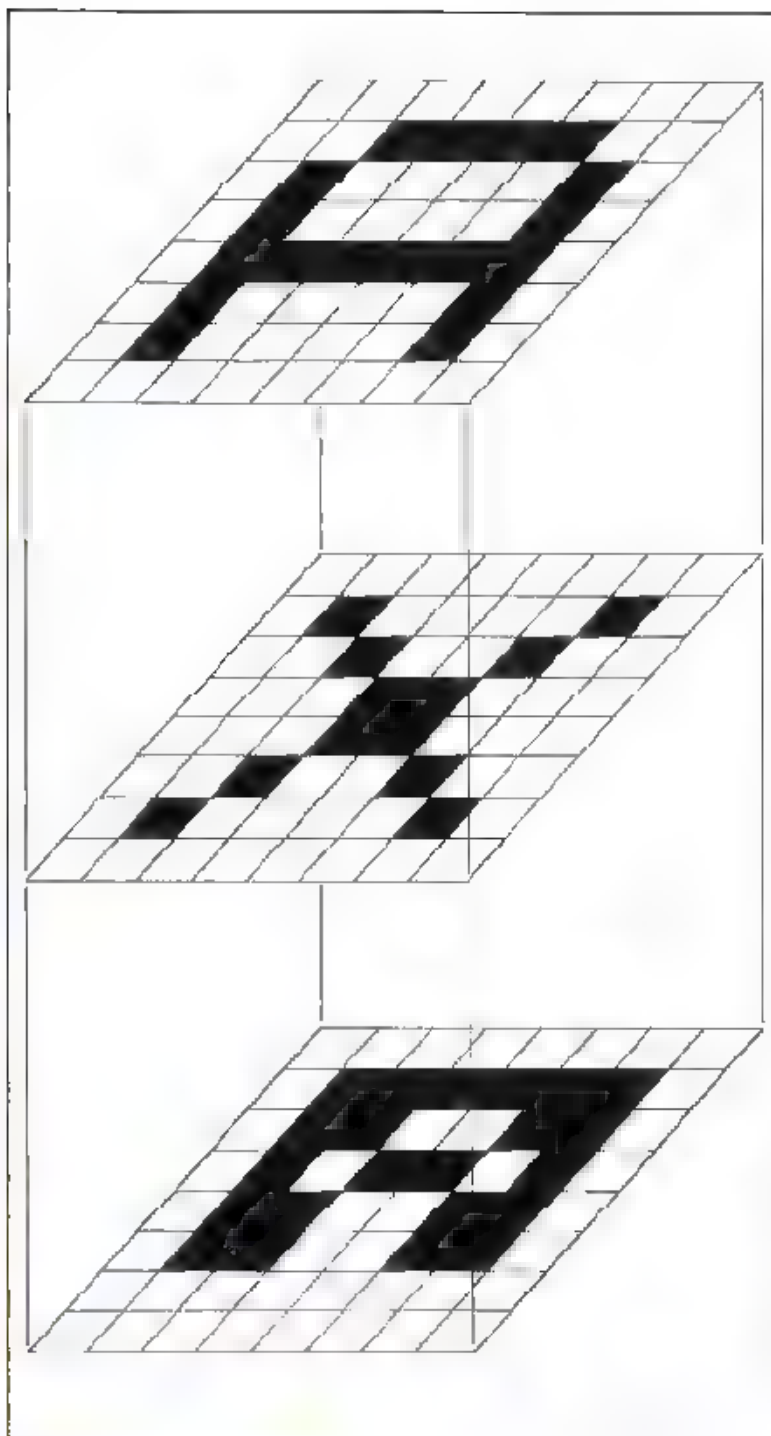
Observe las diferencias en precisión y velocidad.

Este otro genera una serie de circunferencias aleatorias; al superponerse cambian de color los bloques de caracteres afectados.

```

10 REM *****
   :          : REDONDELES :
   :          :          :
20 BORDER 1 PAPER 5 C:L3
22 RANDOMIZE

```



Over 1

```

30 LET x=INT (RND*215)+20
40 LET y=INT (RND*135)+20
50 LET radio=INT (RND*15)+5
60 LET color=INT (RND*8)
70 IF color=5 THEN GO TO 80
80 CIRCLE INK color x y radio
90 GO TO 30

```

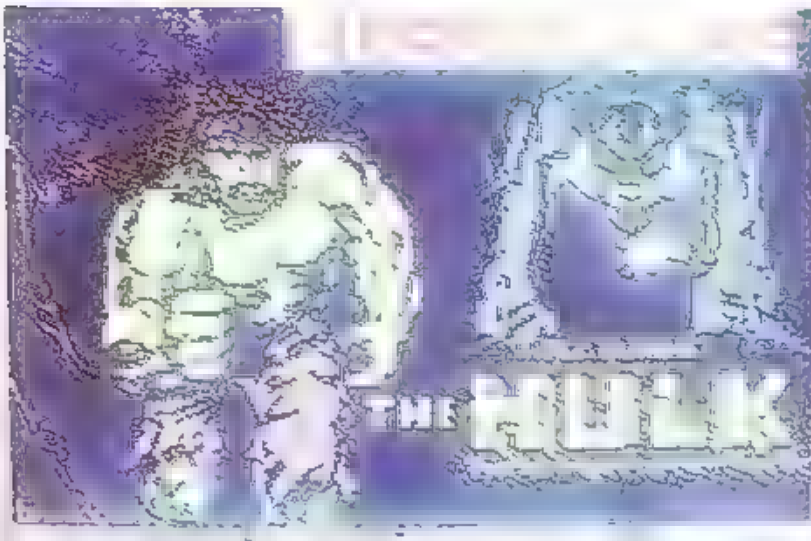
existen otros métodos para realizar gráficos, éstos se basan principalmente en la utilización de

Técnicas avanzadas

Aparte de las sentencias «PLOT», «DRAW» y «CIRCLE»,

Programas
Periféricos

Los creadores de pantallas



Técnicas avanzadas en la realización de gráficos.

gráficas utilizan una serie de programas comerciales, que facilitan su labor a la hora de diseñarlas. Entre los más conocidos están:

- MELBOLRNE DRAW
- PAINTBOX
- SCREEN MACHINE
- ARTIST

Lógicamente, las características de cada uno de estos programas son distintas de las del resto, pero, generalizando, se puede

- Desplazar el cursor en las ocho direcciones, pixel a pixel, sin dibujar.
- Idem dibujando.
- Idem dibujando en inversa.
- Idem borrando.
- Cambiar los atributos de cada bloque de 64 pixel
- Rellenar áreas de un color determinado
- Desplazar un gráfico hasta situarlo en la posición adecuada
- Invertir la pantalla de izquierda a derecha.

- Definir GOU (Gráfico Definido por el Usuario), almacenarlos en cinta y utilizarlos posteriormente
- Incluir textos.
- Algunos permiten ampliar una zona de la pantalla (zoom), gracias a esta opción se puede dibujar con una mayor precisión.

Existen comercialmente ciertos *periféricos*, o dispositivos que se conectan al ordenador, con los que también se pueden realizar gráficos; normalmente necesitan de un software para ser utilizados.

Principalmente existen cuatro tipos:

- LAPIZ OPTICO
- MESA GRAFICA
- PANTALLA DE VIDEO TACTIL
- RATON

El *Lápiz Optico* (Light Pen) es un dispositivo que al desplazarlo por la pantalla como si estuviera escribiendo, detecta la luz emitida por el televisor y la transforma en señales eléctricas, las cuales son enviadas al ordenador para ser procesadas. Una vez conocidas las coordenadas del punto donde está posicionado el lápiz, el programa se encarga de visualizar el correspondiente pixel

La *Mesa Gráfica* es una especie de tablero donde se acopla un *brazo articulado*, conectado mecánicamente con unos componentes conocidos por el nombre de *potenciómetros*. Este tipo de *transductor* envía una información al ordenador, que depende de la posición del brazo.

Para conocer la posición de un punto se utiliza un sistema de *coordenadas polares*.

Existen también otro tipo de mesas más sofisticadas, don-

PROGRAMA 6

```

1 REM *****
  * CURSOR/BASIC *
  * ***** *
  * MAPA *
  * ***** *
2 BORDER 4: PAPER 4: INK 1, C
3
4 RESTORE
10 READ X,Y
15 IF X=0 THEN GO TO 40
20 PLOT X,Y
30 GO TO 10
40 INK 0
45 PLOT 159,7
50 DRAW 96,0
60 DRAW 0,40
70 DRAW -96,0
80 DRAW 0,40
90 PLOT 200,7
100 DRAW 24,40
110 READ X,Y
120 IF X=0 THEN GO TO 150
130 PLOT X,Y
140 GO TO 110
150 PAUSE 0: STOP
5000 REM
*****
* PENINSULA *
* ***** *
5010 DATA 133,175,134,174,134,17
3,134,172,133,171,133,170,133,16
9,132,168
5020 DATA 131,167,130,166,129,16
5,128,164
5030 DATA 127,165,126,164,125,16
3,124,163,123,162,122,163,121,16
4,120,163
5040 DATA 119,163,118,164,117,16
4,116,163,115,164,114,163,113,16
6,112,165
5050 DATA 111,165,110,164,109,16
3,108,164,107,164,106,165,105,16
5,104,165
5060 DATA 103,165,102,165,101,16
6,100,166,99,165,98,166,97,167,9
6,167
5070 DATA 95,166,94,165,93,165,9
2,165,91,165,90,166,89,165,88,16
5
5080 DATA 87,165,86,164,85,165,8
4,165,83,165,82,166,81,166,80,16
7
5090 DATA 79,167,78,168,77,168,76,168,75,168,74,169,73,168,72,16
9
5100 DATA 71,169,70,169,69,169,6
8,170,67,171,66,170,65,171,64,17
0,64,169
5110 DATA 63,168,62,169,61,169,6
0,170,59,169,58,169,57,170,56,17
0
5120 DATA 55,169,54,169,53,168,5
3,167,52,166,51,167,51,168,51,16
9,50,170,49,170,48,169
5130 DATA 47,169,46,170,45,170,4
4,171,43,172,42,173,42,173,41,17
4,40,174,40,173
5140 DATA 39,172,38,173,38,174,3
8,175,37,174,36,173,35,173,34,17
4,35,175,33,173,32,172
5150 DATA 31,173,30,172,29,171,2
8,170,29,169,30,169,31,169,31,16
8
5160 DATA 31,167,30,166,29,165,2
8,166,28,167,27,165,26,166,26,16
7,25,166,24,166
5170 DATA 23,166,22,167,21,166,2
0,167,19,167,18,166,17,166,16,16
5,17,164,16,164
5180 DATA 16,164,14,165,14,163,1
3,163,12,162,13,161,13,160
5190 DATA 12,159,12,158,14,159,1
4,158,14,157,14,156,14,155,15,15
6,15,152

```

```

5200 DATA 15,158,17,155,16,154,1
5,153,16,152,17,152,18,153,19,18
3,20,152
5210 DATA 19,151,19,150,18,149,1
8,149,18,148,20,147,18,146,18,14
6,20,146,21,146,22,146,19,145,22
,145,21,144,15,149
5220 DATA 20,143,19,143,18,142,1
7,141,17,140,17,139,17,138,16,13
7,17,136
5230 DATA 1,135,17,134,17,133,1
8,132,17,131,17,130,18,129,17,12
8
5240 DATA 18,127,18,126,16,125,1
8,124,18,123,18,122,19,121,19,12
0
5250 DATA 19,119,19,118,18,117,1
8,116,18,115,18,114,17,113,19,11
3,13,112
5260 DATA 19,111,19,110,16,109,1
7,108,16,108,16,109,16,110
5270 DATA 15,107,15,106,15,106,1
4,104
5280 DATA 14,103,13,102,13,101,1
4,101,15,101,15,100,15,99,14,99,
13,98,12,97,12,98
5290 DATA 12,95,11,94,11,93,11,9
2,10,91,10,90,9,89,9,88
5300 DATA 8,87,8,86,8,85,7,84,6,
84,8,83,5,82,5,81,5,80
5310 DATA 4,78,4,78,3,77,3,76,3,
75,3,74,3,73,2,72
5320 DATA 2,71,2,70,3,70,4,70,3,
71,6,71,7,71,6,72,9,72,10,73,11,
72
5330 DATA 11,71,10,70,9,70,8,69,
7,68,6,69,5,68,5,67,6,66,6,65,6,
64,7,64
5340 DATA 8,64,9,64,10,65,11,65,
12,66,12,65,13,65,14,66,16,65,15
64
5350 DATA 15,63,15,62,14,62,13,6
3,11,63,12,62,12,61,12,60,12,59,
12,58,12,57,11,56
5360 DATA 11,55,10,54,10,53,11,5
3,11,51,11,50,11,49,11,48
5370 DATA 11,47,11,46,11,45,11,4
4,10,43,10,42,10,41,9,40
5380 DATA 9,39,9,38,8,37,8,36,7,
35,7,34,8,34,9,34,10,35,11,35,12
35,13,35,14,35,15,35
5390 DATA 18,34,17,34,18,34,19,3
4,20,34,21,34,22,33,23,33
5400 DATA 24,32,25,31,26,32,27,3
2,28,33,29,33,30,34,31,34
5410 DATA 32,35,33,35,34,35,35,3
5,36,36,37,36,36,36,34,36
5420 DATA 40,35,41,36,42,35,43,3
5,44,34,45,33,46,32
5430 DATA 47,31,48,30,40,39,50,0
9,51,28,51,27,51,26,50,25,50,24
5440 DATA 51,23,52,23,53,23,54,2
2,55,21,55,20,54,19,53,18,52,19,
52,20,53,18,54,17,54,16
5450 DATA 55,15,56,15,57,14,58,1
3,59,12,60,11,61,11,62,10,63,9
5460 DATA 64,6,65,9,66,10,66,11,
67,12,68,12,68,11,69,13,69,14,70
15
5470 DATA 71,16,71,17,72,18,73,1
8,74,18,75,18,76,19,77,18,78,18,
79,18
5480 DATA 80,19,81,20,82,21,83,2
2,84,22,85,23,85,23,87,22
5490 DATA 86,22,89,23,90,23,91,2
3,92,23,93,24,94,24,95,23
5500 DATA 96,23,97,24,98,24,99,2
4,100,23,101,23,102,23,103,22
5510 DATA 104,23,105,23,106,23,1
07,23,108,24,109,23,110,23,111,2
2
5520 DATA 112,23,113,23,114,23,1
15,23,116,24,117,25,118,25,119,2
5
5530 DATA 120,25,121,24,122,24,1
23,23,124,25,125,25,126,24,126,2
5,126,26,127,27,127,28
5540 DATA 128,28,129,30,130,30,1
29,31,129,32,129,33,130,34,130,3
5,131,36,132,37,133,37,134,37,13
5,38,135,38
5550 DATA 136,40,137,41,138,40,1

```

```

39,40,140 41,141,41,142,40,143,4
5560 DATA 144,41,145,41,146,41,1
47,41,148,42,149,42,148,43,148,4
4,148,45,147,43,146,43,145,44,14
5,45,146,45,146,47
5570 DATA 147,46,147,49,148,50,1
48,51,140,52,149,50,149,54,149,5
5580 DATA 150,56,151,57,151,58,1
52,59,153,60,154,61,155,61,156,6
2,157,62,158,63
5590 DATA 159,64,160,65,161,65,1
62,66,162,67,161,68,160,69,159,6
9,158,69,157,70,156,70,155,71
5600 DATA 155,72,154,73,154,74,1
54,75,164,76,153,77,153,78,153,7
9
5610 DATA 153,80,153,81,153,82,1
53,83,164,84,154,85,154,86,155,8
7
5620 DATA 155,88,155,89,157,90,1
58,91,158,92,158,93,159,94
5630 DATA 160,95,160,96,161,97,1
62,98,162,99,163,100,163,101,164
1,102,165,103
5640 DATA 165,104,166,105,166,10
6,167,107,167,108
5650 DATA 168,108,169,107,169,10
6,168,105,170,108,171,109,170,11
0,169,111
5660 DATA 170,112,171,113,171,11
4,172,115,173,118,174,116,176,11
7
5670 DATA 176,117,177,117,178,11
8,179,118,180,118,181,119,182,11
9,183,119
5680 DATA 184,119,185,120,186,12
0,187,120,188,121,189,121,190,12
1,191,122,191,123
5690 DATA 192,124,193,125,194,12
6,195,127,196,127
5700 DATA 197,128,198,129,199,12
9,200,129,201,130,202,131,203,13
1,204,132,205,133,206,134,207,13
3
5710 DATA 207,136,207,137,207,13
6,207,139,208,140,208,141,208,14
2,207,143
5720 DATA 208,144,208,145,208,14
6,207,146,208,147,208,148,208,14
9,204,150,204,151
5730 DATA 204,152,204,153,204,15
4,204,155,203,156,204,167,203,15
8,203,159
5740 DATA 203,160,204,161,205,16
2,205,163,206,164,207,165
5750 DATA 208,166,209,166,210,16
7,211,168,212,169,213,169,214,17
0,215,171
5760 DATA 216,172,217,173,218,17
3,219,173,220,172,221,171,222,17
1,223,170
5770 DATA 224,171,225,171,224,17
2,225,173,226,173,227,172,228,17
1,227,170,228,169
5780 DATA 225,168,226,167,227,16
7,228,168,229,168,230,167,231,16
7,232,168
5790 DATA 231,169,230,170,231,17
1,232,171,233,170,234,169,235,16
9,236,170,237,170,238,169,239,16
8
5800 DATA 240,167,241,167,242,16
8,243,167,244,165,244,165,245,16
5,246,164
5810 DATA 247,164,240,164,249,16
5,250,164,251,163,252,163,253,16
2,254,162,255,162
5820 REM

```

```

*****
*
* BALEARES
*
*****

```

```

5830 DATA 215,88,214,89,213,90,2
12,89,211,88,210,89,209,90,210,8
1,209,92,210,93,210,94,209,94,20
0,93
5840 DATA 207,93,208,92,208,92,2

```

```

04,91,203,90,202,90,201,89,200,8
8
5850 DATA 199,87,199,86,198,85,1
97,84,199,84
5860 DATA 200,83,201,84,202,84,2
03,84,204,83,204,82,204,81,205,8
1,205,81,207,80
5870 DATA 206,79,204,79,204,77,2
07,76,210,80,211,81,212,82,213,8
3,213,84,213,85,214,85,215,87
5880 DATA 183,72,182,72,181,72,1
81,71,180,70,179,70,180,59,180,6
8,181,69,182,68,183,68,184,68,18
3,69
5890 DATA 184,70,185,71,186,72,1
85,73,184,73
5900 DATA 184,65,183,65,182,64,1
83,63,183,64,184,64,185,64,186,6
4
5910 DATA 224,95,223,97,222,95,2
21,95,220,96,219,95,220,94,220,9
3,221,94,222,94,223,94
5920 DATA 224,93,225,93,225,92,2
27,92,228,93,227,94,226,95,225,9
6
5930 REM

```

```

*****
*
* CANARIAS
*
*****

```

```

5940 DATA 200,24,201,25,202,26,2
03,27,203,28,203,29,202,29,201,2
9,200,28
5950 DATA 100,27,108,27,107,26,1
06,26,195,26,194,26,193,26,192,2
6
5960 DATA 191,24,190,24,189,24,1
88,24,188,23,189,22,190,21,191,2
1
5970 DATA 192,19,193,18,193,17,1
94,16,195,16,196,17,197,18,197,1
9,198,20,198,21,199,22,199,23
5980 DATA 184,17,183,17,182,17,1
81,18,181,19,181,20,182,21,183,2
1,184,20,185,20,186,20,186,19
5990 DATA 169,30,170,31,170,30,1
71,29,171,28,172,27,173,28,174,2
9,174,30,174,31,174,32,174,33,17
3,34,172,34,171,35,170,34,169,34
169,33
6000 DATA 169,30,168,30,168,31,1
69,32,170,32,171,32,172,33,173,3
2,173,31,172,30,171,29,170,30
6010 DATA 216,12,215,12,214,11,2
13,11,212,11,211,12,210,13,210,1
4,210,15
6020 DATA 211,16,211,17,212,16,2
12,19,213,20,214,20,215,19
6030 DATA 216,19,217,20,217,18,2
18,17,218,16,218,15,218,14,217,1
3
6040 DATA 240,24,240,25,240,25,2
41,27,241,28,241,29,242,30,243,3
0,244,30,246,31,245,29,245,28,24
5,27,245,28,244,25,244,24
6050 DATA 244,23,244,22,243,21,2
42,21,241,20,240,20
6060 DATA 239,19,239,18,237,17,2
36,16,235,16,234,17,234,18,235,1
6,236,19,237,20,236,21,239,22,23
9,23
6070 DATA 240,40,241,40,241,44,2
42,40,243,40,243,45,244,44,4
1
6080 DATA 244,39,244,38,243,37,2
42,37,241,36,240,35
6090 DATA 239,36,238,35,237,36,2
38,37,238,38,239,39,0,0
6095 REM

```

```

*****
*
* ESPANA
*
*****

```

```

6100 DATA 1,8,2,8,3,8,4,8,5,8,6,

```

```

8,6,9,2,9,2,10,2,11,2,12,2,13,2,
14,1,14,3,14,4,14,5,14,6,14,10
13,11,14,11,14,12,4,10
6110 DATA 6,9,9,8,10,8,11,8,12,8
13,8,13,9,13,10,13,11,12,11,11,
11,10,11,9,11,12,9,13,9,14,10,
14,11,14,12,14,10,14,13,14
6120 DATA 16,8,17,8,18,8,17,9,17
10,17,11,17,12,17,13,17,14,16,1
4,18,14,19,14,20,14,21,14,21,13,
21,12,21,11,20,11,19,11,10,11
6130 DATA 23,8,24,8,25,8,24,9,24
10,24,11,24,12,24,13,24,14,25,1

```

```

4,26,14,27,14,28,14,28,13,28,12,
28,11,28,10,28,9,28,8,27,8,24,8,
25,11,26,11,27,11
6140 DATA 31,8,32,8,33,8,32,9,32
10,32,11,32,12,13,32,14,31,1
4,33,14,35,8,36,8,37,8,36,9,36,1
8,36,11,36,12,36,10,36,14,36,14,
37,14,33,12,34,11,35,10,33,16,34
16,35,16
6150 DATA 39,8,40,8,41,8,40,9,40
10,40,11,40,12,40,13,40,14,41,1
4,42,14,43,14,44,14,44,13,44,12,
44,11,44,10,44,9,44,8,43,8,45,8,
41,11,42,11,43,11,0,0

```

de el traductor ha sido sustituido por una especie de lápiz y la mesa incorpora un complejo sistema de detección, bien sea por campos eléctricos, magnéticos o ultrasónicos.

Otro tipo de periférico es la pantalla de vídeo sensible al tacto; con ella se puede dibujar directamente con el dedo, posicionándolo en los lugares elegidos.

El sistema de detección, de la zona de la pantalla tocada con el dedo, está basado en la

interrupción de rayos infrarrojos.

Por último, el *ratón* es un pequeño mando que según se va desplazando sobre la mesa, genera un gráfico idéntico al movimiento de éste.

Es un periférico muy fácil de utilizar y además, existe una perfecta coordinación entre el movimiento de la mano sobre la mesa y la observación de la pantalla por parte del usuario.

OVER

Acceso al teclado

IN KEY \$



MODULO E

SYMBOL
SHIFT

OVER

Tipo de sentencia

Control de impresión.

PROGRAMA 7

```

1 REM *****
  * C JRSO/BASIC *
  * *****
  * OVER *
  * *****
2 BORDER 1 PAPER 4 INK 0 C
LS
5 FOR z=1 TO 2
10 OVER 1
20 LET paso=5
30 RANDO 112
40 LET x=INT (RND*256)
50 LET y=INT (RND*176)
60 LET x1=x
70 LET xf=256-x
80 LET y1=y
90 LET yf=176-y
100 FOR n=x1 TO xf STEP paso
102 PLOT x,y
110 DRAW n,yf
120 NEXT n
130 LET xf=256-x
140 LET y1=yf
150 LET yf=-y
160 FOR n=y1-paso TO yf STEP p
170 PLOT x,y

```

```

180 DRAW xf,n
190 NEXT n
200 LET x1=xf
210 LET xf=-x
220 FOR n=x1-paso TO xf STEP -p
230 PLOT x,y
240 DRAW n,yf
250 NEXT n
260 LET y1=yf
270 LET yf=176-y
280 FOR n=y1+paso TO yf-paso ST
290 PLOT x,y
300 DRAW xf,n
310 NEXT n
320 PRINT #0, " Pulsar una tecla
para continuar"
324 PAUSE 0 INPUT 0
330 NEXT z
340 FOR z=1 TO 7
350 FOR n=0 TO 21
360 PRINT PAPER z, INK 9,AT n,0
370 NEXT n
380 PAUSE 200
390 NEXT z
400 PAUSE 0

```


PROGRAMA 8

```

10 REM *****
11 * CURSO/BASIC *
12 * *****
13 * LASERINTO *
14 * *****
15
16 BORDER 1: PAPER 1: INK 7: C
17
18 GO SUB 2000
19 LET detecta=1000
20 LET contador=0
21 LET tiempo=0
22 LET record=100
23 REM INICIO
24 FOR n=1 TO 20
25 PRINT AT 0,1, INVERSE 1, IN
26 K 4: 00000000000000000000000000000000
27 GO
28
29 NEXT n
30 REM INSTRUCCIONES
31 FOR n=5 TO 7
32 PRINT AT n,9, '
33 NEXT n
34 FOR n=10 TO 12
35 PRINT AT n,19, '
36 NEXT n
37 FOR n=15 TO 18
38 PRINT AT n,5, '
39 PRINT AT n,23, '
40 NEXT n
41 REM COMANDO
42 READ y, '
43 IF NOT x THEN GO TO 280
44 PRINT AT y,x, '
45 GO TO 180
46 REM THE FACE CARDS
47 DATA 1,2,2,2,3,2,4,2,5,2,6,
48 7,2,8,2,9,2,10,2,11,2,12,2,13,
49 13,3,13,4,13,5,13,6,14,6,15,6,
50 15,6,28,6
51 DATA 6,3,6,4,6,5,6,6,6,7,8,
52 8,6,12,6,13,6,14,7,14,8,14,9,14,
53 10,14,11,14,12,14,12,15,12,16,11,
54 16,16,16,15,15,15,15,15,15,15,
55 16,4,16,4,16,4,16,3,16,2,16,1,16
56 DATA 17,18,17,9,17,10,17,11,
57 11,19,11,19,12,19,13,19,14,19,
58 15,19,16,19,17,19,17,17,17,17,1
59 17,19,17,20,16,20,15,20,14,20,
60 14,21,14,22,14,23,14,24,15,24
61 DATA 25,26,3,26,3,26,4,26,5,26,
62 5,26,6,26,6,26,7,26,8,26,9,26,
63 9,26,10,26,10,26,11,26,11,30,15,2
64 DATA 27,14,27,15,27,15,27,17,27,
65 17,28
66 DATA 0,0
67 REM DETECTA
68 PRINT FLASH 1, AT 6,10, "X" A
69 11 20, "X", AT 17,6, "X", AT 17,24
70
71 REM FIGURA
72 LET y=INT (RND*22)
73 IF y=0 OR y=21 THEN GO TO 3
74
75 GO TO 310
76 LET x=INT (RND*32)
77 PRINT AT 1,6, '
78 REM DETECTA
79 PRINT PAPER 6 AT 2,5, TIEMPO
80
81 PRINT PAPER 6, AT 14,10, 'SAC
82 contador
83 PRINT PAPER 6, AT 10,5, 'PMAX
84 record
85 REM INICIAR
86 IF INKEY$="7" THEN LET x0=x
87 LET y0=y-1 GO SUB detecta
88 IF INKEY$="6" THEN LET x0=x
89 LET y0=y+1 GO SUB detecta
90 IF INKEY$="8" THEN LET x0=x
91 LET y0=y GO SUB detecta

```

```

410 IF INKEY$="5" THEN LET x0=x
411 LET y0=y GO SUB detecta
412 LET tiempo=tiempo+1
413 LET p=STR$ tiempo
414 PRINT PAPER 6, AT 2 15-LEN p
415 tiempo
416 IF tiempo=250 THEN LET punt
417 o=0+contador GO TO 500
418 IF contador=4 THEN LET punt
419 o=300-tiempo GO TO 500
420 GO TO 370
421 REM FIN
422 PRINT AT y,x, '
423 PRINT AT 0,2, 'HAS OBTENIDO
424 "PUNTOS," PUNTOS"
425 PRINT AT 0,1, OVER 1; PAPER
426 0, INK 7, '
427
428 FOR n=0 TO 100 NEXT n
429 PRINT AT 0,1, '1,4, Mujeres ju
430 gar otra vez"
431 PAUSE 0
432 IF INKEY$="n" OR INKEY$="N"
433 THEN CLS STOP
434 IF INKEY$="s" OR INKEY$="S"
435 THEN GO TO 590
436 GO TO 560
437 INPUT 0
438 IF puntos>record THEN LET r
439 ecord=puntos
440 PRINT PAPER 6, AT 10,10, reco
441 rd
442 PRINT AT 0,1, '
443
444 LET tiempo=0 LET contador=
445 0
446 GO TO 280
447 REM OVER, DETECTA
448 IF y0=0 THEN LET y0=0
449 IF y0>21 THEN LET y0=21
450 IF x0=0 THEN LET x0=0
451 IF x0>31 THEN LET x0=31
452 IF SCREEN$ (y0,x0)="" THEN
453 GO TO 1000
454 IF SCREEN$ (y0,x0)="" THEN
455 LET contador=contador+1 PRINT
456 PAPER 7 AT 14 10, contador
457 PRINT AT y,x, '
458 LET y=y0 LET x=x0
459 PRINT AT y,x, '
460 RETURN
461 REM INSTRUCCIONES
462 PLOT 0,0
463 DRAW 0,175
464 DRAW 255,0
465 DRAW 0,-175
466 DRAW 255,0
467 PRINT INVERSE 1; AT 4,0, '
468 © MICROHOBBY
469 PRINT INVERSE 1; AT 17,0, '
470 PU Arriba, Abajo,
471 2080 PRINT AT 10,11, "LABERINTOS"
472 PRINT #0, AT 1,1, Pulsa una
473 tec.a para continuar
474 PAUSE 0
475 CLS
476 PRINT AT 0,2, "Tienes que re
477 coger los cuatro"
478 PRINT "sacos, que hay en la
479 s habitacio"
480 PRINT "nes, en el menor tie
481 po posible"
482 PRINT "Dispones de un tie
483 po limitado"
484 PRINT "de, 250 Pelicentones"
485
486 PRINT AT 11,11, "CONTROLES"
487 PRINT AT 12,11, '
488 PRINT AT 14,9, '7 - Arriba"
489 PRINT AT 16,9, '6 - Abajo"
490 PRINT AT 18,9, '8 - Derecha"
491 PRINT AT 20,9, '5 - Izquierda
492
493 PRINT #0, AT 1,1, "Pulsa una
494 tecla para comenzar"
495 PAUSE 0
496 INK 0 CLS
497 RETURN

```


Definición

La sentencia «OVER» controla la *sobreimpresión* de los caracteres. Su estructura general es

SENTENCIA	ARGUMENTO
OVER	código de control

Ejemplos

- PRINT OVER 1; «BASIC»
- OVER 0
- OVER c
- PRINT OVER n; 1

Cuando el código de control es «cero», al visualizar un carácter en una posición de la pantalla, se borra el que anteriormente hubiese en esa misma posición. Al conectar el ordenador al código de control por defecto es el «0».

Ejemplo

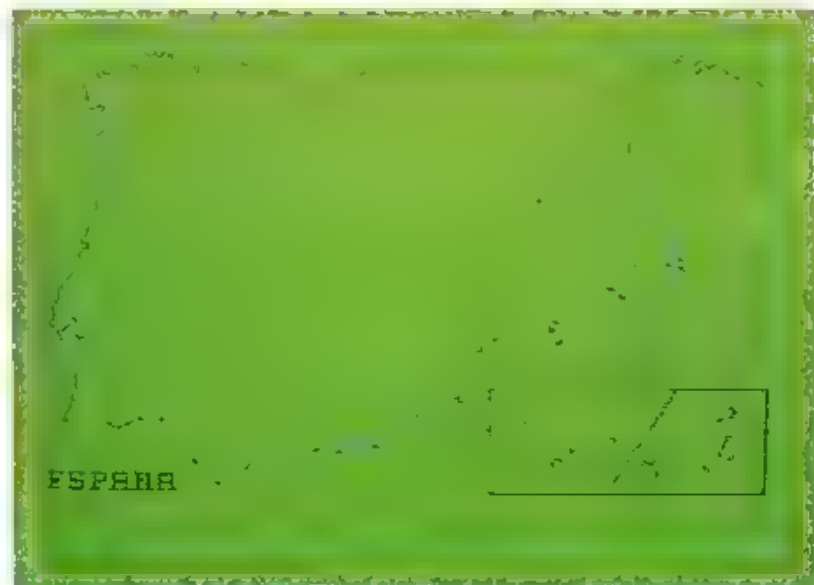
```
10 REM *****
   | OVER 0 |
   |*****|

12 OVER 0
20 PRINT AT 10,10 MICROBOBBY
30 PRINT AT 10,10 BUENAS NOCHES
40 PAUSE 0
50 PRINT AT 10,10 1
60 PRINT AT 10,10 2
```

Con el código de control «uno», se combinan uno a uno, los píxeles del carácter antiguo y del nuevo de acuerdo con la función lógica «XOR» (exclusive OR), cuya *tabla de verdad* es la siguiente

a	1	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Es decir, si los dos píxeles que se combinan tienen color de «papel» o de «tinta» el píxel resultante tendrá color de «papel»; por el contrario, si son distintos, es decir, si uno tiene color de papel y otro de tinta,



el píxel resultante tendrá color de «tinta»

Ejemplo

```
10 REM ++ *****
   | A OVER X |
   |*****|

20 PRINT AT 0,0 A OVER X
30 PRINT AT 0,10 A
40 PRINT AT 0,10 X
```

Si el código de control es distinto de los mencionados anteriormente nos aparecerá

el error de «Invalid colour» o «Integer out of range»

El siguiente programa es una aplicación curiosa de la sentencia «OVER»

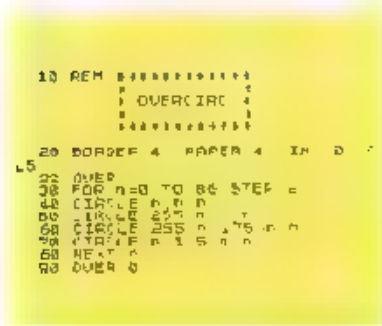
```
10 REM *****
   | OVER |
   |*****|

20 REM @ MICROBOBBY

30 LIST OVER 1
40 PRINT AT INVERSE 1 PAUSE 0
50 LOCATE 10,10 CONT OVER PAUSE 0
60 INPUT 0
70 FOR N=1 TO 3
80 PRINT INVERSE 1 AT 0,10
90 OVER 0
```

Al realizar gráficos con la sentencia «OVER» activada, se obtienen unos resultados artísticos.

El siguiente programa realiza unos círculos crecientes, tomando como origen las cuatro esquinas; aproximadamente tarda dos minutos en completar el gráfico



El programa número «7», con la función «OVER» activada, genera un punto aleatorio, a partir del cual se trazan líneas rectas hacia los bordes de la zona de visualización, posteriormente se genera otro y se repite el proceso, de esta manera se obtiene una combinación «espectacular», el color del fondo va cambiando de color, con una temporización, mientras que la «tinta» tiene color de contraste

SCREEN\$

Acceso al teclado
 MODO **E**
 LEN SYMBOL SHIFT



SCREEN\$
Tipo de sentencia

Función auxiliar

Definición

Esta función retorna una cadena con el carácter existente en una determinada posición. Su estructura general es la siguiente

SENTENCIA	ARGUMENTO
SCREEN\$	(I.B., COLUMNA)

Ejemplo
 — PRINT SCREEN\$ (5,7)
 — IF SCREEN\$ (2, 3) = "9"
 THEN
 — LET a\$ = SCREEN\$ (10, 11)
 — LET b = VAL (SCREEN\$ (7, 4))

«SCREEN\$» reconoce cualquier carácter comprendido entre los códigos 32 (espacio) y el 127 () en decimal independientemente del atributo temporal que tengan

Ejemplo

```
10 PRINT FLASH: 1 AT 10: 10
130
20 PRINT SCREEN$ (10, 11)
```

Esta función retorna una cadena vacía cuando la posición especificada contiene un carácter fuera del rango

```
10 PRINT CHR$ 129
20 PRINT SCREEN$ (0, 0)
```

El programa número «8» genera un laberinto y utiliza la función «SCREEN\$» para detectar los muros y los objetos

Almacenamiento de pantallas

«SCREEN\$» puede utilizarse conjuntamente con las sentencias «SAVE» y «LOAD»

para grabar en cinta y posteriormente cargar la imagen (gráficos + texto) que hay en pantalla

La estructura general es
 a) **Salvar pantallas**

SAVE nombre SCREEN\$

donde el nombre es una cadena entrecomillada de un máximo de diez caracteres
Ejemplo

SAVE "data 10" SCREEN\$

b) **Cargar pantallas**

LOAD nombre SCREEN\$

Ejemplo.

LOAD "data 10" SCREEN\$

La sentencia «VERIFY» no opera conjuntamente con «SCREEN\$», de manera que no puede verificarse la grabación

POINT

Acceso al teclado



Tipo de sentencia

Función auxiliar

Definición

La función «POINT» indica si un píxel es de color de «in-

ta» o de «papel»

Su estructura general es

SIN FUNCIA	ARGUMENTO
POINT	(COORD. X, COORD. Y)

Ejemplos

- PRINT POINT (12, 120)
- LET a = POINT (70, 40)
- IF POINT (5, 3) = 1 THEN
- PRINT POINT (X, Y)

La función «POINT» retorna un «cero» si el pixel especificado tiene color de «papel» y un «uno» si es de color de «tinta»

Ejemplo

a)

PRINT 120
PRINT POINT (10, 12)

b)

PICT 80, 127
PRINT POINT (80, 127)

Programa

La estructura del programa «LABERINTO», el número 8, es la siguiente

- 4 Continúa con el borde de programa
- 12 Asignación de los colores azul y negro y fondo, y dibujo para los caracteres
- 20 Continúa la suma de visualización de mismo
- 22 Asignación a la variable

24 28

40 60

80 160

180 210

220 270

280

310 340

350

354 358

380 410

412

«letra» en el punto de línea donde comienza la salida de la selección

Asignación de los valores de los valores de control de tiempo y de color

Señalando a la pantalla con los colores del fondo y el fondo de la pantalla

Visualización de los valores de los valores de control de tiempo y de color

Borrado de la pantalla y visualización de los valores de los valores de control de tiempo y de color

Trabajo con los datos de los valores de los valores de control de tiempo y de color

Visualización de los valores de los valores de control de tiempo y de color

Generación de la salida de la pantalla en el fondo de los valores de los valores de control de tiempo y de color

Visualización de la salida de la pantalla en el fondo de los valores de los valores de control de tiempo y de color

Visualización de los valores de los valores de control de tiempo y de color

Control de la salida de la pantalla en el fondo de los valores de los valores de control de tiempo y de color

Incremento en la salida de la pantalla en el fondo de los valores de los valores de control de tiempo y de color

14 416

418

419

502

510

520

530

540 580

590

600

610

670

770

1000

1010 1040

1050

1052

1060

1070

1080

2000 2100

2110 2260

del tiempo transcurrido visualización del tiempo transcurrido en la pantalla

Comprobación si ha transcurrido el tiempo de la pantalla

Comprobación si ha transcurrido los cuatro segundos y calcula la puntuación

Borrado de la pantalla y visualización de la pantalla en el fondo de los valores de los valores de control de tiempo y de color

Comprobación si se desea jugar otra partida

Borrado del mensaje visualización de la pantalla en el fondo de los valores de los valores de control de tiempo y de color

Comprobación si los puntos obtenidos superan la puntuación máxima

Visualización de la puntuación obtenida

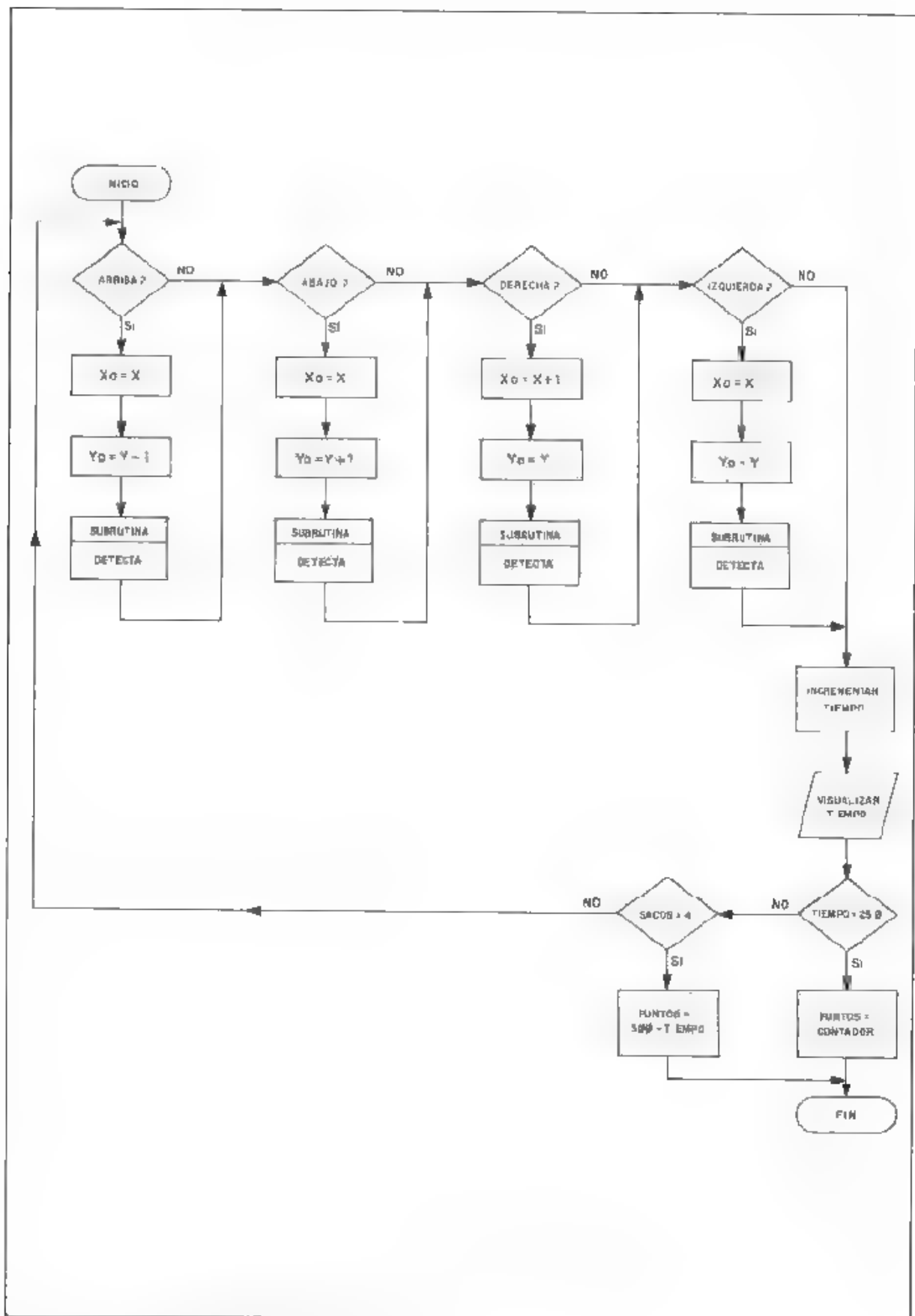
Borrado del mensaje de la pantalla en el fondo de los valores de los valores de control de tiempo y de color

Comprobación si se desea jugar otra partida

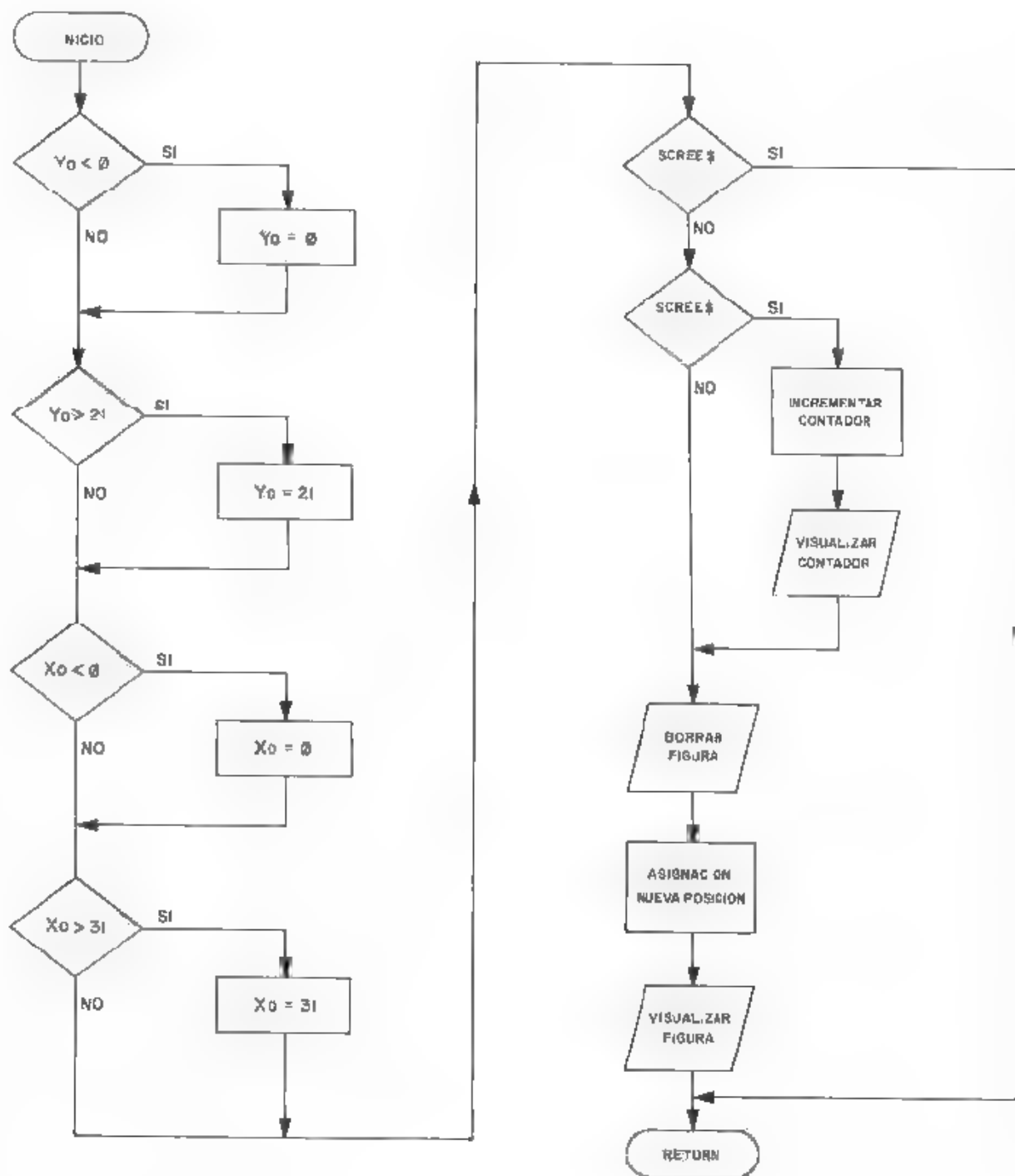
Comprobación si la puntuación es un saco, y aumenta y visualiza el valor de la variable de control de tiempo y de color

Borra la antigua posición y calcula la nueva posición y visualiza la posición de la pantalla en el fondo de los valores de los valores de control de tiempo y de color

Presentación del juego y presentación de las instrucciones



Rutina movimiento. Programa laberinto.



Subrutina detecta. Programa laberinto.

GRAFICOS DEFINIDOS

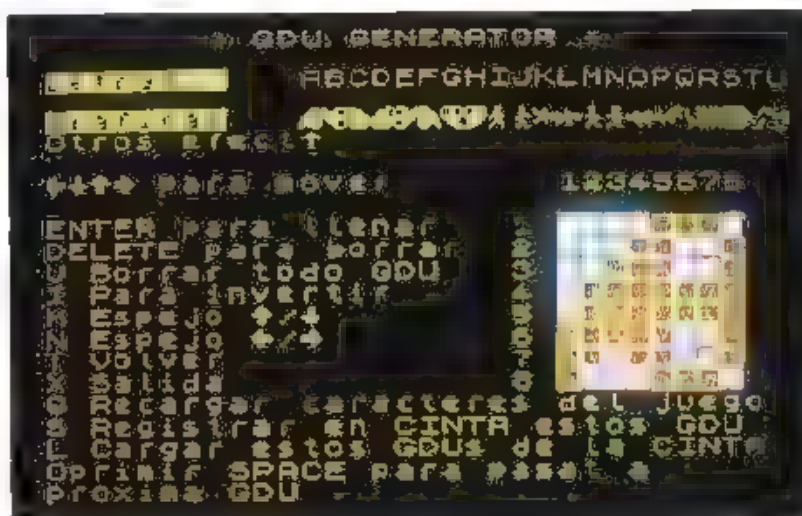
Posiblemente éste sea uno de los capítulos más esperados por el lector, ya que, tras su lectura, le va a permitir definir sin dificultad, sus propios gráficos, con el único límite de su imaginación.

Abreviadamente se le conoce por las siglas inglesas «UDG» (User Defined Graphic) o las españolas «GDU» (Gráficos Definidos por Usuario).

Con las técnicas que se van a describir en este capítulo, se pueden definir hasta 21 gráficos, quedando asignados a las letras comprendidas entre la «A» y la «U», ambas inclusive.

¿Cómo se almacenan?

Al igual que los caracteres cada «GDU» está formado por una matriz de «8» por «8» píxeles, en total 64; el contenido de cada uno de éstos (1-activo, 0-desactivo) queda reflejado en un bit que es la unidad de información más pequeña



utilizada en Informática.

Estos «bits» agrupados de «8» en «8» es lo que se denomina *byte*; en el caso de los gráficos, un «byte» queda formado por os «8» píxeles de una línea; por tanto para almacenar un «GDU» se necesitan ocho «bytes» de información, y para almacenar los 21 posibles gráficos son necesarios 168 «bytes» de información.

Dentro de la memoria del Spectrum, existe una zona

destinada al almacenamiento de los 168 «bytes», esta se encuentra localizada al final de la misma.

Para almacenar un «byte» es necesario indicar en qué posición de memoria debe efectuarse. La sentencia encargada de escribir datos en la memoria es «POKE», su estructura general es:

SENTENCIA	ARGUMENTOS
POKE	dirección, dato

PROGRAMA 1

```

1 REM *****
2 *  C R  E A T I N G  *
3 *  A  L  P  H  A  B  E  T  *
4 *  E  S  P  A  C  I  A  L  *
5 *****
6 PRINT #0 HT 2 : DEFN D
7 : POKE 16383,0
8 : POKE 16384,0
9 : POKE 16385,0
10 : POKE 16386,0
11 : POKE 16387,0
12 : POKE 16388,0
13 : POKE 16389,0
14 : POKE 16390,0
15 : POKE 16391,0
16 : POKE 16392,0
17 : POKE 16393,0
18 : POKE 16394,0
19 : POKE 16395,0
20 : POKE 16396,0
21 : POKE 16397,0
22 : POKE 16398,0
23 : POKE 16399,0
24 : POKE 16400,0
25 : POKE 16401,0
26 : POKE 16402,0
27 : POKE 16403,0
28 : POKE 16404,0
29 : POKE 16405,0
30 : POKE 16406,0
31 : POKE 16407,0
32 : POKE 16408,0
33 : POKE 16409,0
34 : POKE 16410,0
35 : POKE 16411,0
36 : POKE 16412,0
37 : POKE 16413,0
38 : POKE 16414,0
39 : POKE 16415,0
40 : POKE 16416,0
41 : POKE 16417,0
42 : POKE 16418,0
43 : POKE 16419,0
44 : POKE 16420,0
45 : POKE 16421,0
46 : POKE 16422,0
47 : POKE 16423,0
48 : POKE 16424,0
49 : POKE 16425,0
50 : POKE 16426,0
51 : POKE 16427,0
52 : POKE 16428,0
53 : POKE 16429,0
54 : POKE 16430,0
55 : POKE 16431,0
56 : POKE 16432,0
57 : POKE 16433,0
58 : POKE 16434,0
59 : POKE 16435,0
60 : POKE 16436,0
61 : POKE 16437,0
62 : POKE 16438,0
63 : POKE 16439,0
64 : POKE 16440,0
65 : POKE 16441,0
66 : POKE 16442,0
67 : POKE 16443,0
68 : POKE 16444,0
69 : POKE 16445,0
70 : POKE 16446,0
71 : POKE 16447,0
72 : POKE 16448,0
73 : POKE 16449,0
74 : POKE 16450,0
75 : POKE 16451,0
76 : POKE 16452,0
77 : POKE 16453,0
78 : POKE 16454,0
79 : POKE 16455,0
80 : POKE 16456,0
81 : POKE 16457,0
82 : POKE 16458,0
83 : POKE 16459,0
84 : POKE 16460,0
85 : POKE 16461,0
86 : POKE 16462,0
87 : POKE 16463,0
88 : POKE 16464,0
89 : POKE 16465,0
90 : POKE 16466,0
91 : POKE 16467,0
92 : POKE 16468,0
93 : POKE 16469,0
94 : POKE 16470,0
95 : POKE 16471,0
96 : POKE 16472,0
97 : POKE 16473,0
98 : POKE 16474,0
99 : POKE 16475,0
100 : POKE 16476,0
101 : POKE 16477,0
102 : POKE 16478,0
103 : POKE 16479,0
104 : POKE 16480,0
105 : POKE 16481,0
106 : POKE 16482,0
107 : POKE 16483,0
108 : POKE 16484,0
109 : POKE 16485,0
110 : POKE 16486,0
111 : POKE 16487,0
112 : POKE 16488,0
113 : POKE 16489,0
114 : POKE 16490,0
115 : POKE 16491,0
116 : POKE 16492,0
117 : POKE 16493,0
118 : POKE 16494,0
119 : POKE 16495,0
120 : POKE 16496,0
121 : POKE 16497,0
122 : POKE 16498,0
123 : POKE 16499,0
124 : POKE 16500,0
125 : POKE 16501,0
126 : POKE 16502,0
127 : POKE 16503,0
128 : POKE 16504,0
129 : POKE 16505,0
130 : POKE 16506,0
131 : POKE 16507,0
132 : POKE 16508,0
133 : POKE 16509,0
134 : POKE 16510,0
135 : POKE 16511,0
136 : POKE 16512,0
137 : POKE 16513,0
138 : POKE 16514,0
139 : POKE 16515,0
140 : POKE 16516,0
141 : POKE 16517,0
142 : POKE 16518,0
143 : POKE 16519,0
144 : POKE 16520,0
145 : POKE 16521,0
146 : POKE 16522,0
147 : POKE 16523,0
148 : POKE 16524,0
149 : POKE 16525,0
150 : POKE 16526,0
151 : POKE 16527,0
152 : POKE 16528,0
153 : POKE 16529,0
154 : POKE 16530,0
155 : POKE 16531,0
156 : POKE 16532,0
157 : POKE 16533,0
158 : POKE 16534,0
159 : POKE 16535,0
160 : POKE 16536,0
161 : POKE 16537,0
162 : POKE 16538,0
163 : POKE 16539,0
164 : POKE 16540,0
165 : POKE 16541,0
166 : POKE 16542,0
167 : POKE 16543,0
168 : POKE 16544,0
169 : POKE 16545,0
170 : POKE 16546,0
171 : POKE 16547,0
172 : POKE 16548,0
173 : POKE 16549,0
174 : POKE 16550,0
175 : POKE 16551,0
176 : POKE 16552,0
177 : POKE 16553,0
178 : POKE 16554,0
179 : POKE 16555,0
180 : POKE 16556,0
181 : POKE 16557,0
182 : POKE 16558,0
183 : POKE 16559,0
184 : POKE 16560,0
185 : POKE 16561,0
186 : POKE 16562,0
187 : POKE 16563,0
188 : POKE 16564,0
189 : POKE 16565,0
190 : POKE 16566,0
191 : POKE 16567,0
192 : POKE 16568,0
193 : POKE 16569,0
194 : POKE 16570,0
195 : POKE 16571,0
196 : POKE 16572,0
197 : POKE 16573,0
198 : POKE 16574,0
199 : POKE 16575,0
200 : POKE 16576,0
201 : POKE 16577,0
202 : POKE 16578,0
203 : POKE 16579,0
204 : POKE 16580,0
205 : POKE 16581,0
206 : POKE 16582,0
207 : POKE 16583,0
208 : POKE 16584,0
209 : POKE 16585,0
210 : POKE 16586,0
211 : POKE 16587,0
212 : POKE 16588,0
213 : POKE 16589,0
214 : POKE 16590,0
215 : POKE 16591,0
216 : POKE 16592,0
217 : POKE 16593,0
218 : POKE 16594,0
219 : POKE 16595,0
220 : POKE 16596,0
221 : POKE 16597,0
222 : POKE 16598,0
223 : POKE 16599,0
224 : POKE 16600,0
225 : POKE 16601,0
226 : POKE 16602,0
227 : POKE 16603,0
228 : POKE 16604,0
229 : POKE 16605,0
230 : POKE 16606,0
231 : POKE 16607,0
232 : POKE 16608,0
233 : POKE 16609,0
234 : POKE 16610,0
235 : POKE 16611,0
236 : POKE 16612,0
237 : POKE 16613,0
238 : POKE 16614,0
239 : POKE 16615,0
240 : POKE 16616,0
241 : POKE 16617,0
242 : POKE 16618,0
243 : POKE 16619,0
244 : POKE 16620,0
245 : POKE 16621,0
246 : POKE 16622,0
247 : POKE 16623,0
248 : POKE 16624,0
249 : POKE 16625,0
250 : POKE 16626,0
251 : POKE 16627,0
252 : POKE 16628,0
253 : POKE 16629,0
254 : POKE 16630,0
255 : POKE 16631,0
256 : POKE 16632,0
257 : POKE 16633,0
258 : POKE 16634,0
259 : POKE 16635,0
260 : POKE 16636,0
261 : POKE 16637,0
262 : POKE 16638,0
263 : POKE 16639,0
264 : POKE 16640,0
265 : POKE 16641,0
266 : POKE 16642,0
267 : POKE 16643,0
268 : POKE 16644,0
269 : POKE 16645,0
270 : POKE 16646,0
271 : POKE 16647,0
272 : POKE 16648,0
273 : POKE 16649,0
274 : POKE 16650,0
275 : POKE 16651,0
276 : POKE 16652,0
277 : POKE 16653,0
278 : POKE 16654,0
279 : POKE 16655,0
280 : POKE 16656,0
281 : POKE 16657,0
282 : POKE 16658,0
283 : POKE 16659,0
284 : POKE 16660,0
285 : POKE 16661,0
286 : POKE 16662,0
287 : POKE 16663,0
288 : POKE 16664,0
289 : POKE 16665,0
290 : POKE 16666,0
291 : POKE 16667,0
292 : POKE 16668,0
293 : POKE 16669,0
294 : POKE 16670,0
295 : POKE 16671,0
296 : POKE 16672,0
297 : POKE 16673,0
298 : POKE 16674,0
299 : POKE 16675,0
300 : POKE 16676,0
301 : POKE 16677,0
302 : POKE 16678,0
303 : POKE 16679,0
304 : POKE 16680,0
305 : POKE 16681,0
306 : POKE 16682,0
307 : POKE 16683,0
308 : POKE 16684,0
309 : POKE 16685,0
310 : POKE 16686,0
311 : POKE 16687,0
312 : POKE 16688,0
313 : POKE 16689,0
314 : POKE 16690,0
315 : POKE 16691,0
316 : POKE 16692,0
317 : POKE 16693,0
318 : POKE 16694,0
319 : POKE 16695,0
320 : POKE 16696,0
321 : POKE 16697,0
322 : POKE 16698,0
323 : POKE 16699,0
324 : POKE 16700,0
325 : POKE 16701,0
326 : POKE 16702,0
327 : POKE 16703,0
328 : POKE 16704,0
329 : POKE 16705,0
330 : POKE 16706,0
331 : POKE 16707,0
332 : POKE 16708,0
333 : POKE 16709,0
334 : POKE 16710,0
335 : POKE 16711,0
336 : POKE 16712,0
337 : POKE 16713,0
338 : POKE 16714,0
339 : POKE 16715,0
340 : POKE 16716,0
341 : POKE 16717,0
342 : POKE 16718,0
343 : POKE 16719,0
344 : POKE 16720,0
345 : POKE 16721,0
346 : POKE 16722,0
347 : POKE 16723,0
348 : POKE 16724,0
349 : POKE 16725,0
350 : POKE 16726,0
351 : POKE 16727,0
352 : POKE 16728,0
353 : POKE 16729,0
354 : POKE 16730,0
355 : POKE 16731,0
356 : POKE 16732,0
357 : POKE 16733,0
358 : POKE 16734,0
359 : POKE 16735,0
360 : POKE 16736,0
361 : POKE 16737,0
362 : POKE 16738,0
363 : POKE 16739,0
364 : POKE 16740,0
365 : POKE 16741,0
366 : POKE 16742,0
367 : POKE 16743,0
368 : POKE 16744,0
369 : POKE 16745,0
370 : POKE 16746,0
371 : POKE 16747,0
372 : POKE 16748,0
373 : POKE 16749,0
374 : POKE 16750,0
375 : POKE 16751,0
376 : POKE 16752,0
377 : POKE 16753,0
378 : POKE 16754,0
379 : POKE 16755,0
380 : POKE 16756,0
381 : POKE 16757,0
382 : POKE 16758,0
383 : POKE 16759,0
384 : POKE 16760,0
385 : POKE 16761,0
386 : POKE 16762,0
387 : POKE 16763,0
388 : POKE 16764,0
389 : POKE 16765,0
390 : POKE 16766,0
391 : POKE 16767,0
392 : POKE 16768,0
393 : POKE 16769,0
394 : POKE 16770,0
395 : POKE 16771,0
396 : POKE 16772,0
397 : POKE 16773,0
398 : POKE 16774,0
399 : POKE 16775,0
400 : POKE 16776,0
401 : POKE 16777,0
402 : POKE 16778,0
403 : POKE 16779,0
404 : POKE 16780,0
405 : POKE 16781,0
406 : POKE 16782,0
407 : POKE 16783,0
408 : POKE 16784,0
409 : POKE 16785,0
410 : POKE 16786,0
411 : POKE 16787,0
412 : POKE 16788,0
413 : POKE 16789,0
414 : POKE 16790,0
415 : POKE 16791,0
416 : POKE 16792,0
417 : POKE 16793,0
418 : POKE 16794,0
419 : POKE 16795,0
420 : POKE 16796,0
421 : POKE 16797,0
422 : POKE 16798,0
423 : POKE 16799,0
424 : POKE 16800,0
425 : POKE 16801,0
426 : POKE 16802,0
427 : POKE 16803,0
428 : POKE 16804,0
429 : POKE 16805,0
430 : POKE 16806,0
431 : POKE 16807,0
432 : POKE 16808,0
433 : POKE 16809,0
434 : POKE 16810,0
435 : POKE 16811,0
436 : POKE 16812,0
437 : POKE 16813,0
438 : POKE 16814,0
439 : POKE 16815,0
440 : POKE 16816,0
441 : POKE 16817,0
442 : POKE 16818,0
443 : POKE 16819,0
444 : POKE 16820,0
445 : POKE 16821,0
446 : POKE 16822,0
447 : POKE 16823,0
448 : POKE 16824,0
449 : POKE 16825,0
450 : POKE 16826,0
451 : POKE 16827,0
452 : POKE 16828,0
453 : POKE 16829,0
454 : POKE 16830,0
455 : POKE 16831,0
456 : POKE 16832,0
457 : POKE 16833,0
458 : POKE 16834,0
459 : POKE 16835,0
460 : POKE 16836,0
461 : POKE 16837,0
462 : POKE 16838,0
463 : POKE 16839,0
464 : POKE 16840,0
465 : POKE 16841,0
466 : POKE 16842,0
467 : POKE 16843,0
468 : POKE 16844,0
469 : POKE 16845,0
470 : POKE 16846,0
471 : POKE 16847,0
472 : POKE 16848,0
473 : POKE 16849,0
474 : POKE 16850,0
475 : POKE 16851,0
476 : POKE 16852,0
477 : POKE 16853,0
478 : POKE 16854,0
479 : POKE 16855,0
480 : POKE 16856,0
481 : POKE 16857,0
482 : POKE 16858,0
483 : POKE 16859,0
484 : POKE 16860,0
485 : POKE 16861,0
486 : POKE 16862,0
487 : POKE 16863,0
488 : POKE 16864,0
489 : POKE 16865,0
490 : POKE 16866,0
491 : POKE 16867,0
492 : POKE 16868,0
493 : POKE 16869,0
494 : POKE 16870,0
495 : POKE 16871,0
496 : POKE 16872,0
497 : POKE 16873,0
498 : POKE 16874,0
499 : POKE 16875,0
500 : POKE 16876,0
501 : POKE 16877,0
502 : POKE 16878,0
503 : POKE 16879,0
504 : POKE 16880,0
505 : POKE 16881,0
506 : POKE 16882,0
507 : POKE 16883,0
508 : POKE 16884,0
509 : POKE 16885,0
510 : POKE 16886,0
511 : POKE 16887,0
512 : POKE 16888,0
513 : POKE 16889,0
514 : POKE 16890,0
515 : POKE 16891,0
516 : POKE 16892,0
517 : POKE 16893,0
518 : POKE 16894,0
519 : POKE 16895,0
520 : POKE 16896,0
521 : POKE 16897,0
522 : POKE 16898,0
523 : POKE 16899,0
524 : POKE 16900,0
525 : POKE 16901,0
526 : POKE 16902,0
527 : POKE 16903,0
528 : POKE 16904,0
529 : POKE 16905,0
530 : POKE 16906,0
531 : POKE 16907,0
532 : POKE 16908,0
533 : POKE 16909,0
534 : POKE 16910,0
535 : POKE 16911,0
536 : POKE 16912,0
537 : POKE 16913,0
538 : POKE 16914,0
539 : POKE 16915,0
540 : POKE 16916,0
541 : POKE 16917,0
542 : POKE 16918,0
543 : POKE 16919,0
544 : POKE 16920,0
545 : POKE 16921,0
546 : POKE 16922,0
547 : POKE 16923,0
548 : POKE 16924,0
549 : POKE 16925,0
550 : POKE 16926,0
551 : POKE 16927,0
552 : POKE 16928,0
553 : POKE 16929,0
554 : POKE 16930,0
555 : POKE 16931,0
556 : POKE 16932,0
557 : POKE 16933,0
558 : POKE 16934,0
559 : POKE 16935,0
560 : POKE 16936,0
561 : POKE 16937,0
562 : POKE 16938,0
563 : POKE 16939,0
564 : POKE 16940,0
565 : POKE 16941,0
566 : POKE 16942,0
567 : POKE 16943,0
568 : POKE 16944,0
569 : POKE 16945,0
570 : POKE 16946,0
571 : POKE 16947,0
572 : POKE 16948,0
573 : POKE 16949,0
574 : POKE 16950,0
575 : POKE 16951,0
576 : POKE 16952,0
577 : POKE 16953,0
578 : POKE 16954,0
579 : POKE 16955,0
580 : POKE 16956,0
581 : POKE 16957,0
582 : POKE 16958,0
583 : POKE 16959,0
584 : POKE 16960,0
585 : POKE 16961,0
586 : POKE 16962,0
587 : POKE 16963,0
588 : POKE 16964,0
589 : POKE 16965,0
590 : POKE 16966,0
591 : POKE 16967,0
592 : POKE 16968,0
593 : POKE 16969,0
594 : POKE 16970,0
595 : POKE 16971,0
596 : POKE 16972,0
597 : POKE 16973,0
598 : POKE 16974,0
599 : POKE 16975,0
600 : POKE 16976,0
601 : POKE 16977,0
602 : POKE 16978,0
603 : POKE 16979,0
604 : POKE 16980,0
605 : POKE 16981,0
606 : POKE 16982,0
607 : POKE 16983,0
608 : POKE 16984,0
609 : POKE 16985,0
610 : POKE 16986,0
611 : POKE 16987,0
612 : POKE 16988,0
613 : POKE 16989,0
614 : POKE 16990,0
615 : POKE 16991,0
616 : POKE 16992,0
617 : POKE 16993,0
618 : POKE 16994,0
619 : POKE 16995,0
620 : POKE 16996,0
621 : POKE 16997,0
622 : POKE 16998,0
623 : POKE 16999,0
624 : POKE 17000,0
625 : POKE 17001,0
626 : POKE 17002,0
627 : POKE 17003,0
628 : POKE 17004,0
629 : POKE 17005,0
630 : POKE 17006,0
631 : POKE 17007,0
632 : POKE 17008,0
633 : POKE 17009,0
634 : POKE 17010,0
635 : POKE 17011,0
636 : POKE 17012,0
637 : POKE 17013,0
638 : POKE 17014,0
639 : POKE 17015,0
640 : POKE 17016,0
641 : POKE 17017,0
642 : POKE 17018,0
643 : POKE 17019,0
644 : POKE 17020,0
645 : POKE 17021,0
646 : POKE 17022,0
647 : POKE 17023,0
648 : POKE 17024,0
649 : POKE 17025,0
650 : POKE 17026,0
651 : POKE 17027,0
652 : POKE 17028,0
653 : POKE 17029,0
654 : POKE 17030,0
655 : POKE 17031,0
656 : POKE 17032,0
657 : POKE 17033,0
658 : POKE 17034,0
659 : POKE 17035,0
660 : POKE 17036,0
661 : POKE 17037,0
662 : POKE 17038,0
663 : POKE 17039,0
664 : POKE 17040,0
665 : POKE 17041,0
666 : POKE 17042,0
667 : POKE 17043,0
668 : POKE 17044,0
669 : POKE 17045,0
670 : POKE 17046,0
671 : POKE 17047,0
672 : POKE 17048,0
673 : POKE 17049,0
674 : POKE 17050,0
675 : POKE 17051,0
676 : POKE 17052,0
677 : POKE 17053,0
678 : POKE 17054,0
679 : POKE 17055,0
680 : POKE 17056,0
681 : POKE 17057,0
682 : POKE 17058,0
683 : POKE 17059,0
684 : POKE 17060,0
685 : POKE 17061,0
686 : POKE 17062,0
687 : POKE 17063,0
688 : POKE 17064,0
689 : POKE 17065,0
690 : POKE 17066,0
691 : POKE 17067,0
692 : POKE 17068,0
693 : POKE 17069,0
694 : POKE 17070,0
695 : POKE 17071,0
696 : POKE 17072,0
697 : POKE 17073,0
698 : POKE 17074,0
699 : POKE 17075,0
700 : POKE 17076,0
701 : POKE 17077,0
702 : POKE 17078,0
703 : POKE 17079,0
704 : POKE 17080,0
705 : POKE 17081,0
706 : POKE 17082,0
707 : POKE 17083,0
708 : POKE 17084,0
709 : POKE 17085,0
710 : POKE 17086,0
711 : POKE 17087,0
712 : POKE 17088,0
713 : POKE 17089,0
714 : POKE 17090,0
715 : POKE 17091,0
716 : POKE 17092,0
717 : POKE 17093,0
718 : POKE 17094,0
719 : POKE 17095,0
720 : POKE 17096,0
721 : POKE 17097,0
722 : POKE 17098,0
723 : POKE 17099,0
724 : POKE 17100,0
725 : POKE 17101,0
726 : POKE 17102,0
727 : POKE 17103,0
728 : POKE 17104,0
729 : POKE 17105,0
730 : POKE 17106,0
731 : POKE 17107,0
732 : POKE 17108,0
733 : POKE 17109,0
734 : POKE 17110,0
735 : POKE 17111,0
736 : POKE 17112,0
737 : POKE 17113,0
738 : POKE 17114,0
739 : POKE 17115,0
740 : POKE 17116,0
741 : POKE 17117,0
742 : POKE 17118,0
743 : POKE 17119,0
744 : POKE 17120,0
745 : POKE 17121,0
746 : POKE 17122,0
747 : POKE 17123,0
748 : POKE 17124,0
749 : POKE 17125,0
750 : POKE 17126,0
751 : POKE 17127,0
752 : POKE 17128,0
753 : POKE 17129,0
754 : POKE 17130,0
755 : POKE 17131,0
756 : POKE 17132,0
757 : POKE 17133,0
758 : POKE 17134,0
759 : POKE 17135,0
760 : POKE 17136,
```

PROGRAMA 2

```

10 REM *****
   +-----+
   + LINES BASIC +
   +-----+
   + LINES      +
   + GRAFICA    +
   +-----+
20 PRINT AT 0,0 LINES GRAFICA

```

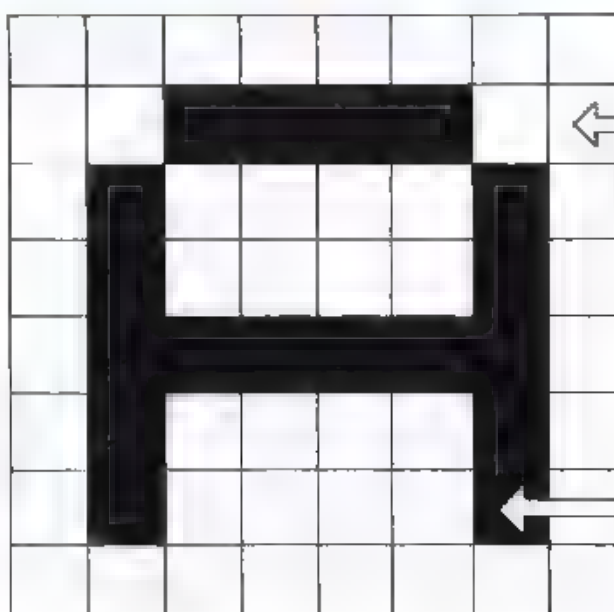
```

10 PRINT AT 1,0
20 FOR I=0 TO 9
30 PRINT CHR$(144+I)
40 PRINT CHR$(144+I)
50 PRINT CHR$(144+I)

60 PRINT HR$(144+I)
70 NEXT I
80 PRINT AT 14,0 CHR$(144)
90 PRINT CHR$(144)
100 PRINT CHR$(144)

```

BYTE 0
BYTE 1
BYTE 2
BYTE 3
BYTE 4
BYTE 5
BYTE 6
BYTE 7



BYT = 0

BYT = 1

Bit y Byte

El siguiente programa nos indica cuál es la dirección de comienzo de cada «GDU».

```

10 REM *****
   +-----+
   + LINES BASIC +
   +-----+
   + LINES      +
   + GRAFICA    +
   +-----+
20 PRINT AT 0,0 LINES GRAFICA

```

Como es lógico, estas direcciones son distintas en los Spectrum de 16 K y 48 K, ya que este último tiene más memoria, por tanto, si para definir el primer «byte» del «GDU» asignado a la letra «A» utilizamos

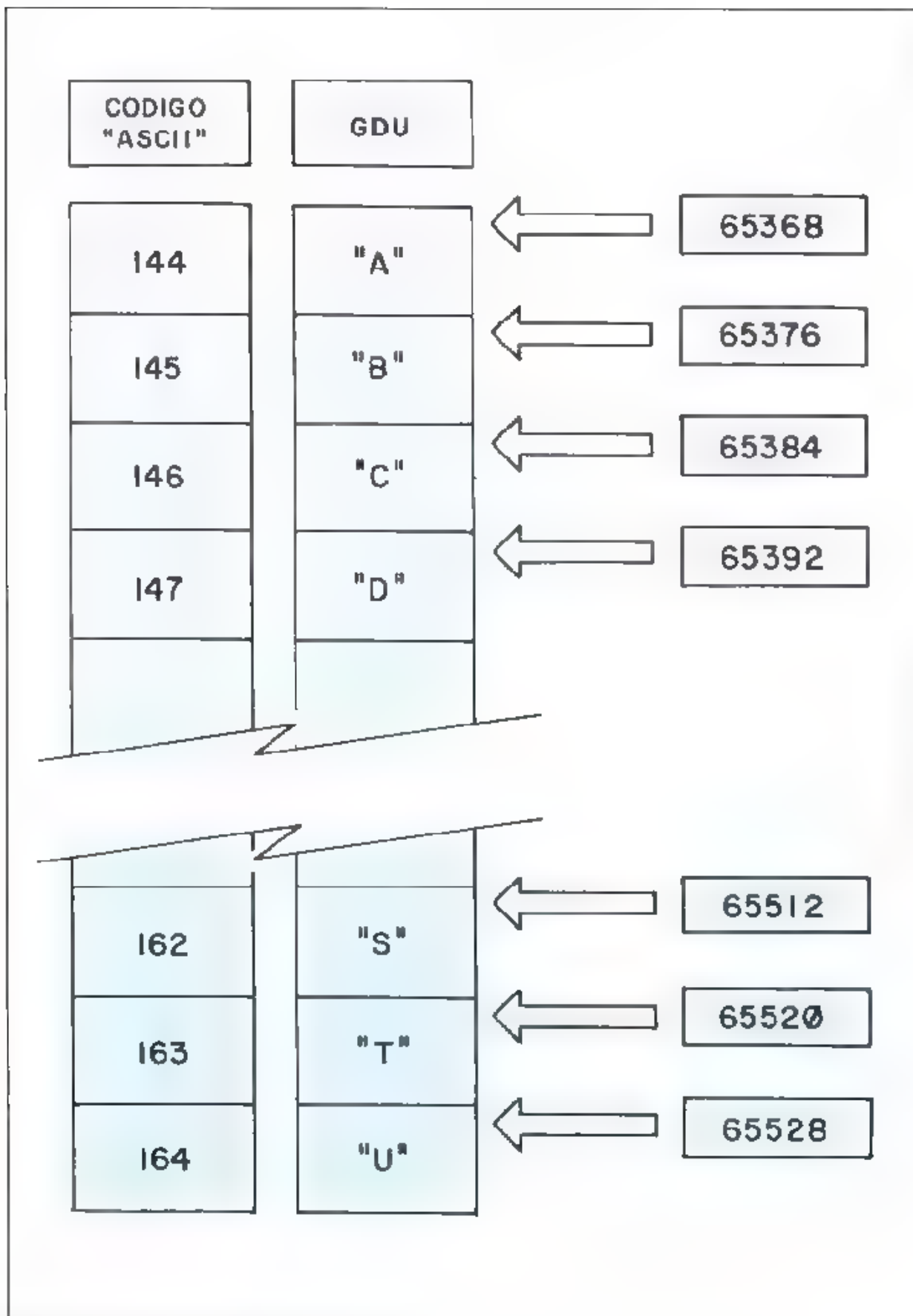
POKE 65538, 144

esta fórmula será correcta para un Spectrum de 48 K,

pero no para uno de 16 K por lo tanto, para almacenar los gráficos debe utilizarse la expresión

POKE 65538, 144

que lo almacena en su posición correspondiente, independientemente de la capacidad de memoria del ordenador. Utilizando esta expresión, el ejemplo anterior que-



Direcciones de comienzo «GDU» (48 K).

¿O SE CUAL ES OBJETO
 DE LAS SUSPIROS ES
 O CONOZCO LA CAUSA DE TU DULCE
 SORRETA LANGUIDEZ,
 TE RIES... ALGUN DIA
 SABRAN RINA POR QUE
 TU LO SABES APENAS
 Y YO NO SE
 "....."
 HOY COME AJER, MAÑANA COMO HOY
 Y SIEMPRE IGUAL

DIOS MIO, QUE SOLOS
 SE QUEDAN LOS MUERTOS

"SE HA PERDIDO" Y DONDE?
 PREGUNTÓ A OCHO INCORPORÁNDOSE
 DESORIENTADO Y CON LAS MANOS
 CRIPITABLE EXPRESIÓN DE TEMOR Y
 ESPERANZA.

RIMAS Y LEYENDAS / G. R. Becerra

Aplicación de los «GDU» en la confección de los textos en español.

NOTAS GRÁFICAS

A B C D E F G H I J

M A < > ± ✱ ✱ ✱ ✱ ✱ ✱

Notas gráficas Programa 4.

PROGRAMA 4

```

1 REM *****
  *
  * CURSO BASIC *
  *
  * *****
  *
  * PALITROQUE *
  *
  * *****
2 PRINT FOR H=1 TO 10:2:ESPE
3A P.C.PAIOB
3 RESTORE RANDOMIZE GO 10
E 9000
4 BORDER 0 PAPER 0 INK 7 C
5 GO 31E 9200
6 REM VARIABLES
7 LET A=11 LET B=11
8 LET S=10=1 LET OBJETO=0
10 LET BORR=0 LET NO_LICHO=2
12 LET S BOR=0 LET PEDADO=0
14 REM SELECCION MINUSCULAS
16 POKS 255=5
60 REM EFECTOS
70 FOR N=0 TO 16 STEP 4
80 PRINT PAPER 6 INK 2 AT N 1
*****
90 NEXT N
100 FOR N=0 TO 16
110 PRINT PAPER 6 INK 2 AT N 0
120 AT N 31
120 NEXT N
122 PRINT PAPER 6 INK 2 AT 1 4
  
```

64389	195
64390	35
64391	126
64392	219
64393	255
64394	65
64395	126
64396	219

GRAFICO "F": ✱

Ejemplo Programa 3.

PERSONAJES

✱ Palitroque
 ✱ Trogloditas

OBJETOS

✱ Espada Gloriosa
 ✱ Escudo Protector
 ✱ Corona Real
 ✱ Nectar de los Dioses
 ✱ Llaves

Pantalla de información Programa 4.

```

1. AT 2 4 " AT 3 4 INVERSE
124 PRINT PAPER 6 INK 2 AT 5 4
" AT 6 4, " AT 7 4. INVERSE
1.
125 PRINT PAPER 6 INK 2 AT 9 2
" AT 10 27 " AT 11 27 IN
VERSE 1 "
126 PRINT PAPER 6 INK 2 AT 13
27 " AT 14 27 " AT 15 27 IN
VERSE 1 "
128 REM RESCALEM
130 LET Y=15
132 LET X=INT (RND*7)+2
140 GO SUB 6000
150 LET Y=11
  
```



```

8204 LET salto=1
8210 RETURN
8220 REM SUB. SALTO
8230 IF ATTR (yf+1,xf)=50 THEN
8240 PRINT OVER 1, BRIGHT 1 AT yf,xf
8250 LET yf=yf+salto LET xf=xf
8260 PRINT OVER 1, INK 4, AT yf,xf
8270 IF ATTR (yf+1,xf)=50 OR ATTR (yf,xf+salto)=50 THEN GO SUB 8100
8280 RETURN
8290 GO SUB 8500
8300 PRINT OVER 1, BRIGHT 1, AT yf,xf
8310 LET yf=yf+salto LET xf=xf+salto
8320 PRINT OVER 1, INK 4, AT yf,xf
8330 IF ATTR (yf+1,xf)=50 OR ATTR (yf,xf+salto)=50 THEN GO SUB 8100
8340 RETURN
8350 GO SUB 8500
8360 PRINT OVER 1, BRIGHT 1 AT yf,xf
8370 LET yf=yf+salto LET xf=xf+salto
8380 PRINT OVER 1, INK 4, AT yf,xf
8390 IF ATTR (yf+1,xf)=50 OR ATTR (yf,xf+salto)=50 THEN GO SUB 8100
8400 RETURN
8410 LET yf=yf+salto LET xf=xf+salto
8420 PRINT OVER 1, INK 4, AT yf,xf
8430 RETURN
8440 REM SUB. REC. LLAVES
8450 IF ATTR (yf+1,xf)=55 THEN PRINT AT yf+1,xf+salto, "GO TO 8600"
8460 IF ATTR (yf,xf+salto)=55 THEN PRINT AT yf,xf+salto, "GO TO 8600"
8470 IF ATTR (yf-1,xf+salto)=57 THEN PRINT AT yf-1,xf+salto, "GO TO 8620"
8480 IF ATTR (yf,xf+salto)=57 THEN PRINT AT yf,xf+salto, "GO TO 8620"
8490 IF ATTR (yf+1,xf+salto)=59 THEN PRINT AT yf+1,xf+salto, "GO TO 8630"
8500 IF ATTR (yf,xf+salto)=59 THEN PRINT AT yf,xf+salto, "GO TO 8630"
8510 RETURN
8520 PRINT FLASH 1, PAPER 6, INK 4, AT 15,27, "RETURN"
8530 PRINT FLASH 1, PAPER 6, INK 4, AT 11,27, "RETURN"
8540 PRINT FLASH 1, PAPER 6, INK 4, AT 7,27, "RETURN"
8550 PRINT FLASH 1, PAPER 6, INK 4, AT 3,27, "RETURN"
8560 REM SUB. GRAFICOS
8570 FOR n=USR 3 TO USR 4
8580 READ dato
8590 POKE n,dato
8600 NEXT n
8610 DATA 252,252,252 0,231,231,231
8620 DATA 126,66,66,66,126,66,66,66

```

```

9060 DATA 4 10 17 10 20 32,80,32
9070 DATA 0 10 4 10 15 32,64 0
9080 DATA 0 24 60 60 24,60 60 0
9090 DATA 195,36 126 219,255 66
9100 DATA 24 88,126 26,24,60,36,102
9110 DATA 0,0,0,165,163,90,126,126
9120 DATA 124 124,124,124 56,16,16 124
9130 DATA 12 119,119 65,119 119 54,28
9140 RETURN
9150 REM GRABATULA
9160 PLOT 0 0
9170 DRAW 0,175 DRAW 255 0
9180 DRAW 0 -175 DRAW -255 0
9190 RETURN
MICROBASIC
9200 PRINT AT 7,11 "PALITROQUE"
9210 AT 13,13, "LOS AT 13 10 TROG"
9220 ITOS
9230 PRINT #0 AT 1,0 "Que es la"
9240 "trocada?" PAUSE 0
9250 IF INKEY$="" THEN CLS RE
9260 IF INKEY$="" THEN GO TO 9300
9270 GO TO 9280
9280 CLS
9290 PRINT AT 0,2 "Palitroque e"
9300 "s el signo de la"
9310 "ordenada, es el"
9320 "para recoger las arm"
9330 "as y objetos"
9340 "Gobernando"
9350 "las Cruzadas."
9360 PRINT "La mision de Palit"
9370 "roque no es"
9380 "las habitacio-"
9390 "can de radas."
9400 PRINT "Palitroque debera"
9410 "recoger las"
9420 "idrac con los"
9430 "citos."
9440 PRINT #0, AT 1,1, "Pulsa una"
9450 "tecla para continuar" PAUSE 0
9460 CLS
9470 PRINT AT 0,4, "PERSONAJES"
9480 PRINT AT 1,4, "4"
9490 PRINT BRIGHT 1, INK 1 AT 3,4 "Palitroque"
9500 PRINT BRIGHT 1 INK 6 AT 5,4 "Trogilios"
9510 PRINT AT 3,4, "OBJETOS"
9520 PRINT AT 3,4, "4"
9530 PRINT BRIGHT 1, INK 2 AT 11,4 "Espada Gloriosa"
9540 PRINT BRIGHT 1, INK 3, AT 13,4 "Escudo Protector"
9550 PRINT BRIGHT 1, INK 4, AT 15,4 "Corona Real"
9560 PRINT BRIGHT 1, INK 5, AT 17,4 "Nectar de los Dioses"
9570 PRINT BRIGHT 1, INK 6 AT 19,4 "Llaves"
9580 PRINT #0 AT 1,1 "Pulsa una"
9590 "tecla para continuar" PAUSE 0
9600 CLS
9610 PRINT AT 0,4 "CONTROLES"
9620 PRINT AT 1,4, "0 - IZQUIERDA"
9630 PRINT AT 3,4, "0 - DERECHA"
9640 PRINT AT 5,4, "0 - SUBIR"
9650 PRINT AT 7,4, "0 - BAJAR"
9660 PRINT AT 9,4, "0 - SALIR"
9670 PRINT AT 11,4, "0 - Para el"
9680 "Juego"
9690 PRINT AT 13,4, "0 - Continua"
9700 PRINT #0, AT 1,1 "Pulsa una"
9710 "tecla para comenzar" PAUSE 0
9720 CLS RETURN

```

sar la tecla «0»

— Para retornar al modo anterior, pulsa la tecla «9».

También pueden utilizarse los «GDU», haciendo referencia a su correspondiente código «ASCII». Por ejemplo, para visualizar el gráfico asignado a la letra «A» utiliza:

```
PRINT CHR$ 144
```

para utilizar otro gráfico consulte la tabla de la página 41.

Al estar situada la zona de memoria de los «GDU» por encima de una *variable del sistema* conocida por el nombre de «RAMTOP», a ejecutar una sentencia del tipo «NEW» se borra la zona de memoria destinada para almacenar nuestro programa BASIC, pero en cambio, permanecen *inalterables* nuestros gráficos, a no ser que desconectemos el ordenador o hagamos un RESET

Programas de aplicación

El programa numero «1» nos permite conocer cuales son los GDU que tenemos almacenados en ese momento en el ordenador y a qué letras están asignados.

El programa «2» genera como GDU una serie de letras y símbolos utilizados en el idioma español que no existen en el teclado del Spectrum, como por ejemplo, la «ñ», la apertura de interrogación «¿», la u con diéresis «ü», etc. Al pulsar una tecla se ejecuta la sentencia NEW que nos borra el programa, pero nos respeta la zona de memoria de los GDU podemos comprobarlo pasando a modo  y pulsando cualquier tecla de la «A» a la «J».

Esta aplicación nos permite

confeccionar textos en español

Grabación de GDU

Podemos grabar en cinta la zona de memoria donde están almacenados los GDU, de esta manera podemos utilizarlos en otra ocasión sin necesidad de tenerlos que definir de nuevo.

La estructura de la sentencia «SAVE» es algo distinta de la utilizada en la grabación de programas, ya que debe especificarse en este caso la dirección de memoria a partir de la cual se desea grabar, así como la longitud en «bytes»:

```
SAVE (nombre) CODE comienzo,  
longitud.
```

La palabra clave «CODE» identifica que no es la zona de memoria donde está almacenado el programa lo que se desea grabar, sino la relacionada en los parámetros «comienzo» y «longitud»

Ejemplo

```
SAVE (gdu1) CODE JSR (a), 168
```

De esta manera, se almacenarán en cinta los 21 posibles GDU. Si por el contrario deseáramos grabar los GDU correspondientes a las letras «C» a «J», ambos inclusive, utilizaríamos

```
SAVE (gdu2) CODE USR (a), 56
```

ya que 56 es el resultado de multiplicar 7 gráficos por 8 bytes cada uno

Para realizar el proceso inverso, es decir, almacenar en memoria los gráficos grabados en cinta, podemos utilizar cualquiera de estas opciones:

nes:

a) Si el siguiente programa a leer, grabado con «CODE» es el especificado:

```
NAN "" CODE
```

b) Especificando el nombre:

```
LOAD (gdu2) CODE
```

c) Especificando la dirección de comienzo:

```
LOAD (gdu2) CODE JSR (a)
```

Este método es más correcto ya que es independiente de la cantidad de memoria que posea el ordenador, por lo tanto él calcula la nueva dirección de carga

d) Especificando también la longitud:

```
LOAD (gdu2) CODE USR (a), 56
```

Lectura de los GDU

El programa numero «3» nos visualiza en la pantalla las direcciones de cada «byte» de un GDU, así como su contenido.

El correspondiente gráfico nos aparece en la parte inferior de la pantalla.

Programa generador de GDU

En las cintas demostración que acompañan tanto al Spectrum 16 o 48 K como el Plus, vienen grabados unos programas con los que se pueden generar con facilidad los GDU. Estos cuentan con una serie de opciones que permiten generar, modificar,

grabar o leer cualquier GDU, y asignarlo a la letra que se desea.

Programa

El programa número «4», que da fin a este capítulo, es una aplicación de los GDU a los juegos.

Las instrucciones son sencillas:

El personaje principal se maneja con las siguientes teclas:

Q - Izquierda
P - Derecha
O - Subir
A - Salto

Hay otras dos opciones que permiten parar el juego o continuar:

S - Para el juego
C - Continúa

La misión de «Palitroque», que es el personaje principal, consiste en recoger los diversos objetos que se encuentran diseminados por las habitaciones, pero no es tan sencilla, ya que las habitaciones se encuentran cerradas. Deberá recoger las llaves que se encuentran suspendidas del techo y tener cuidado con unos bichos llamados «Troglocitos», que le impedirán el paso.

«Palitroque» no dispone de ningún arma, el único modo de esquivar a los terribles «Troglocitos» es saltar sobre ellos, pero deberá tener cuidado de no pisarles y no chocarse con las paredes ya que cualquier golpe eliminará una de nuestros cuatro vidas.

Para pasar de un piso a otro disponemos de una escalera

por la que se puede subir pero no bajar.

Si conseguimos realizar la misión, el rey «Gumersindo» premiará a «Palitroque» con su mayor tesoro.

La estructura del programa es la siguiente:

1 - Comentario con el nombre de programa.

2 - Mensaje de espera para el usuario.

3 - Realiza las siguientes tareas:
- Inicialización de las tablas de datos.
- Inicialización de la función aleatoria.

4 - Llamada a la subrutina que define los GDU.

5 - Asignación de los colores negro con borde y papel, y blanco para los caracteres. Llamada a la subrutina que visualiza las instrucciones.

6-12 - Inicialización de las variables utilizadas en el juego.

13-15 - Utilización de la sentencia «POKE» para inicializar la variable del sistema «FLAG2» localizada en la dirección 23858 con el valor 0, de esta forma se selecciona el modo minúsculas y se simplifica la tarea de detección de teclas pulsadas («KEY\$»).

16-90 - Visualización de los techos y suelos del castillo.

100-120 - Visualización de las paredes.

122-128 - Visualización de las habitaciones y de las cerraduras.

130-180 - Cálculo aleatorio de la posición «X» de la escalera y llamada a la subrutina de visualización.

200-270 - Cálculo aleatorio de la coordenada «X» de la clave y visualización de la misma.

290

310

320

340-390

400-410

507

510

512

520

527

530

540

550

560

570

580

590

Visualización de los objetos dentro de las habitaciones.

Asignación a las variables «X» e «Y» de la posición inicial de «Palitroque».

Visualización de «Palitroque».

Visualización del recuadro inferior que sirve para el contador de vidas y de objetos.

Visualización del número de vidas y de objetos.

Inicialización de las variables «pasos» e incremento utilizadas para el movimiento de los «Troglocitos».

Visualización de los «Troglocitos» en las diversas pisos.

Comprueba si el «Troglocito» se encuentra situado en la posición próxima a «Palitroque», en caso afirmativo se realiza un salto a la subrutina «atrapa» que entre otras cosas nos resta una vida.

Comprueba si se pulsa la tecla «Q» (subir) y salta a la subrutina «subir».

Si no se ha completado el ciclo de subida por la escalera, el programa continúa en la línea 600.

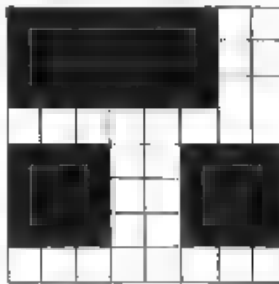
Comprueba si se pulsa la tecla «P» (derecha), calcula la nueva posición de «Palitroque», indica a la variable «salto» que si se realiza un salto será a la derecha, realiza una llamada a la subrutina «derecha» y comprueba si se han recogido los cuatro objetos.

Idem con la tecla «O» (izquierda).

Comprueba si se pulsa la tecla «A» (saltar) para ir a la subrutina «salto».

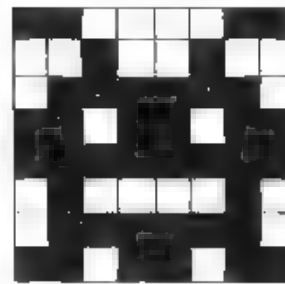
Si la tecla pulsada es la «S» (parada) se queda en un

GDU "A"



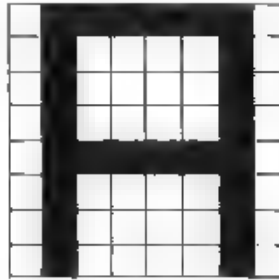
252
252
252
0
231
231
231
0

GDU "F"



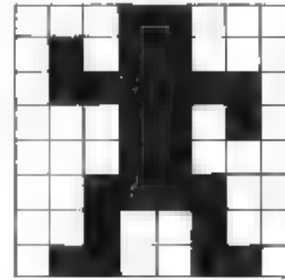
195
36
126
219
255
66
126
29

GDU "B"



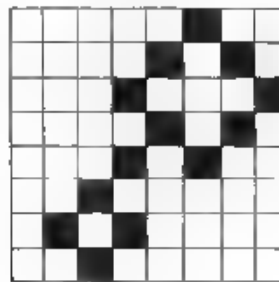
126
66
66
66
126
66
66
66

GDU "G"



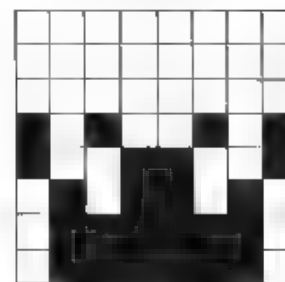
24
88
126
26
24
60
36
102

GDU "C"



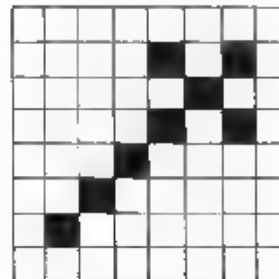
4
10
17
10
20
32
80
32

GDU "H"



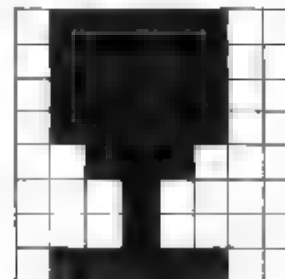
0
0
0
165
153
90
126
126

GDU "D"



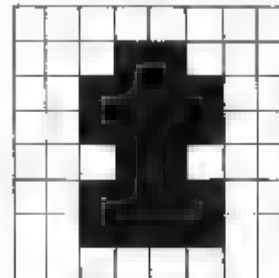
0
10
4
10
6
32
64
0

GDU "I"



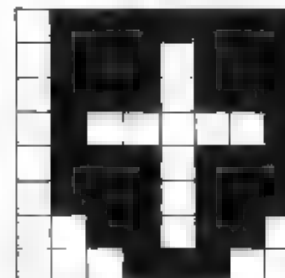
124
124
124
124
56
16
16
124

GDU "E"



0
24
60
60
24
60
60
0

GDU "J"



127
119
119
65
119
119
54
28

«GDU» del Programa «Palitroque y los Trogloditos».

	bucle esperando que se pudiese ir hacia el (continuar)	7120	Idem (escudo protección)	8190	Inicio de la posición de origen de (Palitroque)
6110	Borrado de los (Troglocios)	7130	Idem (corona real)	8200	Visualiza a (Palitroque)
6111	Incremento de la posición de los (Troglocios)	7140	Idem (ectar de los dedos)	8202-8204	Inicialización de las variables
620 630	Calcula si el próximo movimiento de los (Troglocios) debe ser a la derecha o a la izquierda	7160	Aumenta en una unidad el contador de objetos recogidos.	8220	Comienzo de la subrutina salto
640	Salto a la visualización de la nueva posición de los (Troglocios)	7200	Comienzo de la subrutina subir	8222 8430	Comprueba si durante el salto (Palitroque) choca con el muro, es atrapado por algún (Troglocio) o recoge alguna llave
1000	Fin del juego.	7210 7212	Comprueba si (Palitroque) se encuentra en la escalera	8500	Comienzo de la subrutina de recogida de llaves.
1002 1040	Efectos de color en el borde de la pantalla.	7230	Comprueba si al salir de la escalera tropieza con un (Troglocio)	8510 8520	Comprueban que se recoge la llave del piso inferior
1050	Borrado de la pantalla	7232	Incrementa el valor de la variable que almacena el número de peldaños subidos	8530-8540	Idem. del piso primero
1060	Comprueba por que motivo se ha terminado el juego.	7240 7260	Borra y visualiza la nueva posición de (Palitroque) en la escalera.	8550-8560	Idem. del piso segundo.
1070	Mensaje en caso de haberse quedado (Palitroque) sin vidas	7270	Comprueba si (Palitroque) ha terminado de subir la escalera	8570-8580	Idem. del piso superior
1090	Mensaje en caso de haberse completado la misión.	8000	Comienzo de la subrutina que dibuja la escalera	8600	Parpadeo de la cerradura del piso inferior
1100-1130	Comprueba si se desea jugar otra vez o no.	8010	Dibuja la escalera a partir de las coordenadas (x e y)	8610	Idem. del piso primero.
7000	Comienzo de la subrutina (detecta).	8100	Comienzo de la subrutina (atrapa)	8620	Idem. del piso segunda.
7010	Comprueba si la nueva posición está libre o es una escalera.	8110-8140	Parpadeo de (Palitroque) al chocar con la pared o al ser atrapado por algún (Troglocio)	8630	Idem. del piso superior
7022	Comprueba si la puerta de la habitación está abierta	8150	Desaparición del (Palitroque)	9000	Comienzo de la subrutina que genera los gráficos.
7024	Llamada a la subrutina de coger objetos.	8152 8160	Borrado de una vida en el contador de la parte inferior de la pantalla	9002 9030	Bucle para la lectura y generación de los GDI.
7030 7050	Borra la antigua posición de (Palitroque) y visualiza la nueva.	8170	Decremento de una vida.	9040 9130	Tabla con los datos de las GDI
7100	Comienzo de la subrutina de coger objetos.	8180	Comprueba si (Palitroque) se ha quedado sin vidas.	9100	Subrutina (Caratula a las instrucciones)
7110	Comprueba si el objeto cogido es la espada gladi-			9210-9230	Recuadro de la pantalla.
				9240-9260	Mensajes de pantalla.
				9270-9300	Comprueba si se desea visualizar las instrucciones.
				9320-9380	Primera pantalla de información.
				9380-9490	Segunda pantalla de información
				9510 9590	Última pantalla de información.

SONIDO

El Spectrum tiene capacidad para producir una amplia variedad de notas musicales, con el manejo de una sola instrucción

Los sonidos son escuchados a través de un altavoz interno, por lo tanto, el volumen es relativamente bajo, aunque puede mejorarse con el empleo de ciertos periféricos.

En un ordenador personal esta característica es muy apreciada, ya que da vida a ciertos programas que en un principio podrían parecernos «sosos».

Introduzca en el programa del capítulo anterior, «Paltroque y los trogloditas», las siguientes instrucciones y observe las diferencias:

a) Movimiento de los Trogloditos.

```
511 BEEP 0.01, -12
602 BEEP 0.01, 0
```

b, Pérdida de una vida.

```
8122 BEEP 0.01, 12
8132 BEEP 0.01, 24
```

c) Recogida de una llave
Antes de la sentencia «RETURN» de las líneas de programa.

```
0600
8610
0620
8630
```

inclurr

258 MICROBASIC



Amplificador de sonido.

VARIABLE	DIRECCION	VALOR INICIAL
RASP	23608	64
PIP	23609	0

RASP: Zumbador de alarma
PIP: Chasquido del teclado

Tabla 1.

```
BEEP 0.01 48
```

d) Fin de juego

```
1012 BEEP 0.01, z
1014 BEEP 0.01, n
```

```
BEEP
```

Acceso al teclado

LN

MODO E



BEEP

SYMBOL SHIFT

PROGRAMA 1

```

10 REM *****
  *  CURSO/BASIC  *
  * *****      *
  *   HAVIDAD     *
  * *****      *
20 FOR c=-12 TO 0 STEP 12
30 READ nota,duracion
32 BORDER 7
40 IF NOT nota THEN RESTORE
PAUSE 50 GO TO 70
50 BEEP duracion/3,nota+c
52 BORDER 2
60 GO TO 30
70 NEXT c
1000 REM *****

```

```

1010 DATA 16,4,17,1,16,1,15,1
1020 DATA 16,1,17,1,18,1,19,3
1030 DATA 21,2,23,1,24,1,26,1
1040 DATA 24,1,23,1,21,1,19,8
1050 DATA 12,1,14,1,16,2,16,2
1060 DATA 16,1,21,2,19,1,12,2
1070 DATA 12,2,12,1,19,2,17,1
1080 DATA 16,4,17,1,16,1,14,1
1090 DATA 12,1,14,8,16,4,17,1
1100 DATA 16,1,15,1,16,1,17,4
1110 DATA 18,1,19,3,21,2,23,1
1120 DATA 24,1,26,1,24,1,23,1
1130 DATA 21,1,19,5,12,1,14,1
1140 DATA 16,2,16,2,16,1,21,2
1150 DATA 19,1,24,6,12,1,14,1
1160 DATA 16,2,16,2,21,1,5,11,0.
5
1170 DATA 11,1,11,1,12,7 0,0

```

PROGRAMA 2

```

10 REM *****
  *  CURSO/BASIC  *
  * *****      *
  *   DIAFONICA    *
  * *****      *
20 POKE 23658,0
22 GO SUB 1000
30 LET duracion=0.25
40 LET tono=0
50 LET tono=(12 AND INKEY$="s")
+ (4 AND INKEY$="d") + (5 AND INKEY$

```

```

$="f" + (7 AND INKEY$="g") + (9 AND
INKEY$="h") + (11 AND INKEY$="j")
60 IF tono=0 AND INKEY$<>"a" T
HEN GO TO 50
70 BEEP duracion tono
80 GO TO 50
1000 REM *****
1010 PRINT AT 0,6,"ESCALA DIAFON
ICA"
1020 PRINT AT 10,2,"do re mi
fa sol la si"
1030 PRINT AT 12,3,"a s
f"
1040 PRINT AT 1,2,"Toca tu me
lodia preferida"
1050 RETURN

```

Definición

La sentencia «BEEP» activa el altavoz, de manera que suene a una frecuencia proporcional al valor de *semitono* introducido y durante un tiempo determinado.

Su estructura general es

SENTENCIA	ARGUMENTO
BEEP	duración, tono

Ejemplos

- BEEP 1, 3
- BEEP 0.5, 2
- BEEP 10, nota
- BEEP j, a*2

El parámetro «duración» especifica el tiempo en segundos que dura el tono.

Este valor puede estar comprendido dentro de los márgenes: 0 a 10

El valor de tono viene expresado en unidades de semitono, teniendo éstas una relación con el «do» central de un piano.

El valor del semitono es positivo si está por encima del «do» central y es negativo si está por debajo.

Los valores que pueda tomar el parámetro «tono», deben estar comprendidos dentro de los márgenes: -60 a 69.8.

Si se especifican otros valores, tanto por la duración como por el tono, se produce el error

8 integer out of range

Introduzca el siguiente programa que le proporcionará todos los valores tonales que es capaz de reproducir el

Spectrum, primero en una escala creciente y posteriormente, en una decreciente:

```

10 REM *****
  *  ESCALAS TONALES  *
  * *****      *
20 FOR n= 60 TO 69
30 BEEP .05 n
40 NEXT n
50 PAUSE 50
60 FOR n=69 TO -60 STEP -1
70 BEEP .05 n
80 NEXT n

```

Durante la ejecución de una sentencia «BEEP» no puede realizarse «BREAK» ya que el sistema operativo no explorará dicha tecla hasta que no termine de ejecutarse; compruébelo con el comando directo

BEEP 10 0

Edite y ejecute el programa número «1» que interpreta un conocido tema navideño.



Sintetizador de voz.

```

10 REM *****
      *                                     *
      *   CURSO/BASIC                     *
      *                                     *
*****
      *   GRAMATICA                      *
      *                                     *
*****

80 POKE 20000,0
92 GO SUB 1000
30 LET duracao=0.25
40 LET leno=0
50 LET toho=i1 AND INKEY$=""
+ (2 AND INKEY$="s")+ (3 AND INKEY$="z")+ (4 AND INKEY$="d")+ (5 AND
INKEY$="f")+ (6 AND INKEY$="t")+ (7 AND INKEY$="g")+ (8 AND INKEY$="y")+ (9 AND INKEY$="h")+ (10 AND

```

```

INKEY$="U":{11 AND INKEY$=" "
60 IF tone=0 AND INKEY$<"a" T
HEN GO TO 50
70 BEEP duracion,tone
80 GO TO 50
1000 REM INT-4000000
1010 PRINT AT 0,8,"ESCALA CROMAT
ICA"
1020 PRINT AT 12,0,"do re mi
fa sol la si"
1030 PRINT AT 14,1,"a s d
f g h"
1040 PRINT AT 8,2,"1000 1000
1000 1000
1000 1000
1000 1000"
1050 PRINT AT 10,3,"w e
f u"
1060 PRINT AT 1,2,"Toca tu me
lodia preferida"
1070 RETURN

```

```

1 REM          * * * * *
          CURSO/BASIC
          * * * * *
          DOS CRUCES
          *
          * * * * *
10 BORDER 2 PAPER 2: CLS
12 LET duration=0.8
14 LET escala=12
20 REM SELECCION DE DATOS
30 LET do=escala
40 LET dos=escala+1
50 LET re=escala+2
60 LET re5=escala+3
70 LET a1=escala+4
80 LET fa=escala+5
90 LET fa5=escala+5
100 LET sol=escala+7
110 LET sol5=escala+8
120 LET la=escala+9
130 LET la5=escala+10
140 LET si=escala+11
150 REM SELECCION DE DATOS

```

```

160 LET negra=duracion
170 LET redonda=negra*4
180 LET blanca=negra*2
190 LET corchea=negra/2
200 LET semicorchea=negra/4
210 REM FIN DE PROCESO
220 RESTORE
230 READ nota,duracion
232 BORDER 1
240 IF NOT nota AND NOT duracion
  THEN BORDER 2: PAUSE 0: STOP
250 DEEP duracion,nota
252 BORDER 6
260 GO TO 230
270 REM FIN DE PROCESO
272 DATA 0, corchea, re, corchea,
reS, corchea, la, negra, sol, corchea,
faS, corchea, sol, negra, re, blanca,
laS, corchea, laS, corchea, laS, neg
ra, la, blanca, sol, corchea, faS, cor
chea, la, negra, sol, blanca
280 DATA re, corchea, re, corchea,
re, corchea, re, negra, re, corchea, d
o, corchea, re, blanca
290 DATA re, corchea, re, corchea,
re, corchea, fa, negra, reS, corchea,
re, corchea

```



```

300 DATA re3,negra,re,blanca,so
l,corchea,sol,corchea,sol,corche
a,sol,negra,sol,corchea,sol,corc
hea
310 DATA sol,blanca,la,corchea,
la,corchea,la,corchea,la,negra,l
a,corchea,la,corchea,la5,negra,l
a,blanca
320 DATA re,corchea,re,corchea,
re,corchea,re,corchea,re,corchea,d
o,corchea,re,blanca
330 DATA re,corchea,re,corchea,
re,corchea,la,negra,re5,corchea,
re,corchea,re5,negra,re,blanca
340 DATA do,corchea,re,corchea,
re5,corchea,la,negra,sol,corchea,
fa5,corchea,sol,negra,re,negra
350 DATA la5,corchea,la5,corche
a,la5,negra,la,blanca,sol,corche
a,fa5,corchea,la,negra,sol,blanc
a
360 DATA re,corchea,sol,corchea,
sol,corchea,sol,negra,sol,corch
ea,si,corchea,do+12,negra,do+12,
blanca
370 DATA si,corchea,do+12,corch

```

```

ea,do+12,corchea,si,corchea,do+1
2,corchea,sol,corchea,la5,semito
rchea,sol5,corchea,sol,blanca
380 DATA fa5,corchea,fa5,corche
a,fa5,corchea,fa5,negra,mi,corch
ea,re,corchea,mi,negra,re,blanca
390 DATA fa5,corchea,fa5,corche
a,fa5,corchea,mi,corchea,mi,corc
hea,re,corchea,mi,negra,re,blanc
a
400 DATA re,corchea,sol,corchea,
sol,corchea,sol,negra,sol,corch
ea,si,corchea,do+12,negra,do+12,
blanca
410 DATA si,corchea,do+12,corch
ea,do+12,corchea,si,corchea,do+1
2,corchea,sol,corchea,la5,semito
rchea,sol5,corchea,sol,blanca
420 DATA fa5,corchea,fa5,corche
a,fa5,corchea,fa5,negra,mi,corch
ea,re,corchea,mi,negra,re,blanca
430 DATA do,corchea,re,corchea,
mi,corchea,la,negra,sol,corchea,
fa5,corchea,la,negra,sol,blanca
440 DATA 0,0

```

NOTAS		VALOR DE LOS SEMITONOS										
C	DO	-60	-48	-36	-24	-12	0	12	24	36	48	60
C#	DO#	-59	-47	-35	-23	-11	1	13	25	37	49	61
D	RE	-58	-46	-34	-22	-10	2	14	26	38	50	62
D#	RE#	-57	-45	-33	-21	-9	3	15	27	39	51	63
E	MI	-56	-44	-32	-20	-8	4	16	28	40	52	64
F	FA	-55	-43	-31	-19	-7	5	17	29	41	53	65
F#	FA#	-54	-42	-30	-18	-6	6	18	30	42	54	66
G	SOL	-53	-41	-29	-17	-5	7	19	31	43	55	67
G#	SOL#	-52	-40	-28	-16	-4	8	20	32	44	56	68
A	LA	-51	-39	-27	-15	-3	9	21	33	45	57	69
A#	LA#	-50	-38	-26	-14	-2	10	22	34	46	58	
B	SI	-49	-37	-25	-13	-1	11	23	35	47	59	

SOSTENIDO

OCTAVA CENTRAL

Correspondencia entre «notas» y «semitonos» de las diferentes octavas.

Nociones musicales

En este párrafo se van a explicar una serie de conceptos básicos útiles para aquellas personas que deseen programar una melodía en el

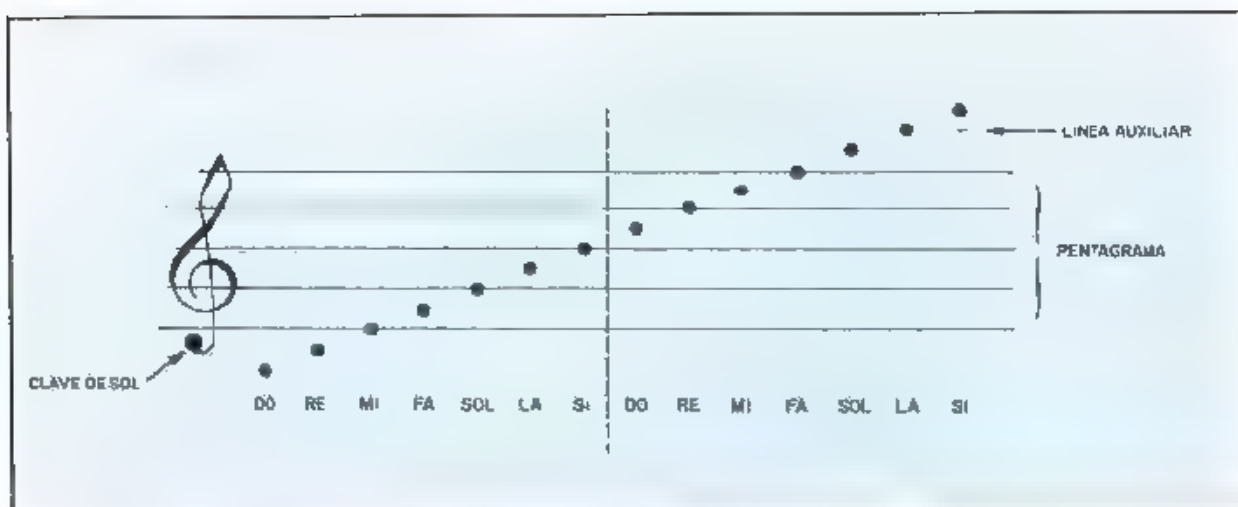
Spectrum.

Tono

El tono es el grado de elevación del sonido, conocido también como *altura tonal*. Siendo el tono más alto cuan-

do mayor es su *frecuencia* (tono agudo) y más bajo cuando es menor (tono grave).

En una escala *diatónica*, las notas musicales identificadoras del tono son.



Escala diatónica de dos octavas, en clave de sol.

DO
RE
MI
FA
SOL
LA
SI

En cambio en la escala cromática son doce:

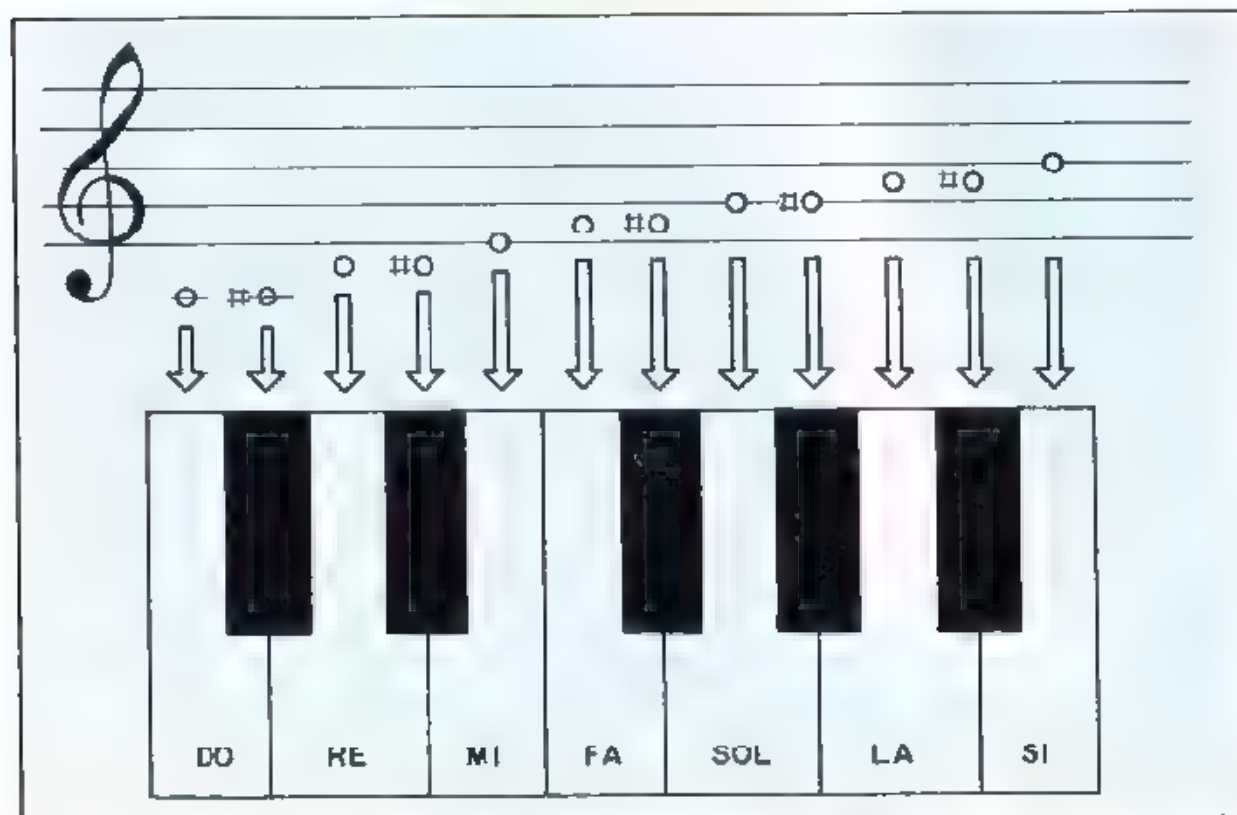
DO #
DO #
RE #
RE #
MI #
FA #
FA #
SOL #
SOL #
LA #
LA #
SI #

El símbolo # es el identificador del sostenido.

En un piano las teclas negras son los correspondientes a los sostenidos.

La unidad máxima de altura tonal es el *semitono*. Un semitono es el resultado de dividir una octava en doce partes iguales.

Entre dos notas consecuti-

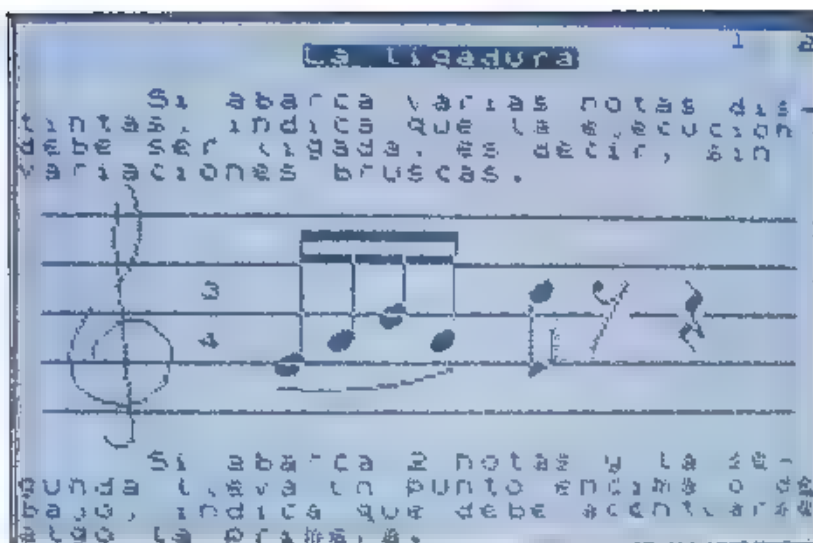


Escala cromática de una octava y su correspondencia con un piano.

vas hay dos semitonos, excepto entre «Mi» - «Fa» y «Si» - «Do» que en ambos casos hay un solo semitono.

En un pentagrama cada nota viene identificada por la posición que ocupa dentro del mismo.

El programa número «2» convierte el teclado del Spectrum en un piano con escala diatónica y el número «3», en un cromático. Las teclas que deben pulsarse en cada caso están contenidas en las instrucciones del propio programa.



Curso de teoría de la música.

PROGRAMA 5

```
10 REM *****
*          *
*  CURSO/BASIC  *
*          *
*****
*  EFECTO 1    *
*          *
*****

20 FOR a=30 TO 0 STEP -1
30 BEEP .005,1
40 NEXT a
```

PROGRAMA 6

```
10 REM *****
*          *
*  CURSO/BASIC  *
*          *
*****
*  EFECTO 2    *
*          *
*****

20 FOR a=30 TO 0 STEP 1
30 BEEP .05,20-a
40 NEXT a
50 BEEP .5,-10
```

PROGRAMA 7

```
10 REM *****
*          *
*  CURSO/BASIC  *
*          *
*****
*  EFECTO 3    *
*          *
*****

20 FOR a=1 TO 25
30 FOR b=1 TO 10
40 BEEP 0.05,INT (RND*b) +12
50 BEEP 0.05,INT (RND*b) +12
60 NEXT b
70 NEXT a
```

PROGRAMA 8

```
10 REM *****
*          *
*  CURSO/BASIC  *
*          *
*****
*  EFECTO 4    *
*          *
*****

20 FOR a=1 TO 10
30 FOR b=10 TO 1 STEP -1
40 BEEP 0.05,a
50 BEEP 0.05,b
60 NEXT b
70 NEXT a
```

Duración

La duración de la nota viene identificada por unos símbolos característicos. Los más utilizados son

REDONDA
BLANCA
NEGRA
CORCHEA
SEM CORCHEA

La que más dura es la redonda, la blanca es la mitad que la redonda, la negra es la mitad que la blanca y así sucesivamente.

Por tanto, en cuanto a dura-

PROGRAMA 9

```
10 REM *****
*          *
*  CURSO/BASIC  *
*          *
* *****
*          *
*    NAVIDAD    *
*          *
*   (TREMOLQ)   *
*          *
* *****
```

```
20 FOR c=-12 TO 0 STEP 12
30 READ nota,duracion
40 IF NOT nota THEN RESTORE .
PAUSE 50 GO TO 70
42 LET j=duracion/0.12
44 FOR x=1 TO j
50 BEEP 0.01,nota+c
52 BORDER 2
54 BEEP 0.01,nota+c+12
56 BORDER 7
58 NEXT x
60 GO TO 30
70 NEXT c
```

ción relativa se refiere, cuatro negras equivalen a una redonda, a dos blancas, a ocho corcheas o a dieciséis semi-corcheas.

Compás

El compás es la separación entre unidades rítmicas, y viene determinado por dos números; el primero, situado en la parte superior del pentagrama, indica el número de partes de un compás y el segundo, situado en la parte inferior, la duración de una de las partes de acuerdo con la siguiente tabla:

NÚMERO	DURACIÓN
2	Blanca
4	Negra
8	Corchea

Ejemplo.

En un compás «dos por cuatro», es decir, compuesto por dos partes y cada una con una duración equivalente a una negra, podrían componer el compás cualquiera de las siguientes combinaciones:

- Una blanca
- Dos negras
- Cuatro corcheas
- Una negra y dos corcheas
- Etc.

Utilizando la técnica del programa número «4», para



Teclado musical simulado.

definir las notas y su duración relativa, puede programar cualquier melodía a través de un pentagrama.

Si desea modificar la duración de las notas o la escala, puede hacerlo variando el contenido de las variables «duración» y «escala», teniendo en cuenta que esta última debe tener un valor múltiplo de doce, bien positivo o negativo.

Variables relacionadas

Dentro de las variables del sistema existen dos relacionadas con el sonido, la primera, denominada «RASP», se encuentra localizada en la dirección 23608 y controla la duración del zumbador de

alarma, esta se activa, por ejemplo, cuando intentamos editar una sentencia de más de 22 líneas.

La otra variable, denominada «PIP» y localizada en la dirección 23609, controla la duración del chasquido del teclado al pulsar una tecla.

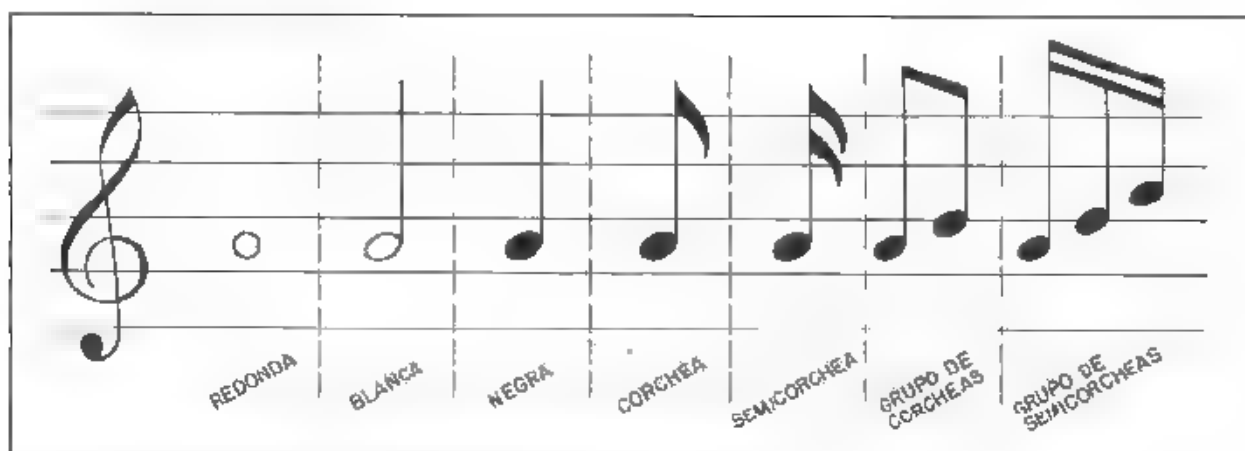
Introduzca el siguiente comando directo y observe la diferencia al pulsar cualquier tecla.

```
POKE 23609 30
```

Estas dos variables pueden tener cualquier valor comprendido entre 0 y 255.

Grabación de sonidos

Si conecta un grabador de



Simbología utilizada en la duración relativa de las notas.

INDICATIVO
DEL COMPAS

COMPAS

DOS POR CUATRO

TRES POR CUATRO

CUATRO POR CUATRO

Ejemplos típicos de compases.

cassette en la salida «MIC» del Spectrum podrá grabar cualquier melodía que genere el ordenador, ya que la señal que activa el altavoz también

se encuentra presente en dicho conector.

También es posible conectar un amplificador a dicha salida, con lo que aumentará

el volumen de la melodía. Si conecta unos auriculares podrá escuchar simultáneamente los sonidos que el ordenador genere, aunque hay

que tener en cuenta que el altavoz interno no se desconecta

Periféricos

Hay ciertos periféricos o dispositivos que podemos conectar al ordenador y que aumentan las posibilidades sonoras de éste.

— Amplificador de sonido que permite escuchar el sonido generado por el ordenador, a través del altavoz de nuestro receptor de T.V., de esta manera se dispone de un control de volumen a gusto del usuario

— Sintetizador de voz que permite al ordenador «hablar» de la misma manera que lo haría un robot. Los sonidos a

emitir deben ser asignados a cierta variable de cadena.

Software musical

Aunque no abundantes, existen en el mercado algunos programas musicales, unos son del tipo didáctico y permiten seguir un curso de «teoría de la música» a través del ordenador Otro, conocido por el nombre comercial de «Melodian», permite asistir a todas las fases de una composición musical, permitiendo

Escribir cartulas
- Efectuar las
modificaciones
- Almacenar en cinta

El programa viene acompañado de una plantilla especial de cartulina que convierte al Spectrum en un teclado musical simulado

Efectos sonoros

Los programas «5» a «8» proporcionan diversos efectos sonoros que pueden ser utilizados en nuestros programas.

El programa numero «9» proporciona un efecto de «TREMOLO», debe ser utilizado con la tabla de datos del programa numero «1», (Navidad).

SENTENCIAS DE GRABACION Y CARGA

El capítulo número tres, "Almacenamiento de programas", fue una breve introducción a la grabación y recuperación de programas BASIC, este va a ser más completo, ya que analizará todos los posibles argumentos de las sentencias

```
SAVE
VERIFY
LOAD
MERGE
```

```
SAVE
```

Acceso al teclado

RESTORE



MODO **K**

Definición

La sentencia "SAVE" permite almacenar en cinta cualquier programa escrito en lenguaje BASIC o en "Código Máquina", así como las matrices

Si estructura general es:

SENTENCIA	ARGUMENTO
SAVE	"nombre" i po

Ejemplo

- SAVE "carg"
- SAVE "CR" CODE 204, 20
- SAVE "T01/2" LNE 40
- SAVE "Pat" DATA Q ()

El campo "nombre" debe ir entre comillas y no superar la cantidad de diez caracteres, estos pueden ser letras, números o símbolos.

Ejemplos:

```
"Trigon"
EMULADOR"
"1234-a"
"$*,?!"
```

A diferencia de las variables, un nombre de programa escrito en minúsculas es distinto de uno en mayúsculas, por tanto los nombres:

```
"COMPILADOR"
"Compiador"
"ComPilador"
```

se refieren a distintos programas.

El campo "tipo" indica qué clase de grabación desea realizar, cuando este se omite, el ordenador interpreta que es la zona de memoria donde está almacenado nuestro programa "BASIC", la que se desea grabar.

Ejemplo:

```
SAVE "ON, OF"
```

De esta manera el programa BASIC denominado "ON/

OF" quedará almacenado en cinta.

Una forma particular de grabar un programa BASIC, es indicar el número de línea donde se desea que comience la *autoejecución*, así, cuando se realice una carga posterior, no será necesario pulsar "RUN" n, donde "n" es el número de línea.

Ejemplo

```
SAVE "ON/OFF" LINE 1000
```

El programa "ON/OFF" se autoejecutará a partir de la línea 1000. Si se desea que un programa se autoejecute a partir de la primera sentencia, puede grabarse con LINE 1.

Grabación de código máquina

Para grabar C/M (Código Máquina), es decir, los programas escritos en el lenguaje interno del ordenador, se utiliza el tipo "CODE", donde es necesario indicar la dirección de memoria a partir de la cual se desea grabar y la longitud del programa expresada en bytes.

Ejemplo:

```
SAVE "S.O." CODE 0 16384
```

de esta forma se almacena en cinta una copia del Sistema operativo del Spectrum, que comienza en la dirección "0"

CINTA: MICROHOBBY-4				CARA: A	
N	Tipo	Nombre	Comienzo	Long.	
1	Prog	MICROHOBBY	1	545	
2	Byte	CASSETTE	32768	6912	
3	Prog	GARDEN	1	237	
4	Byte	garden\$	32768	6912	
5	Byte	GARDEN1	65368	168	
6	Byte	GARDEN2	32492	40	
7	Prog	GARDEN3	1	8244	
8	Prog	SOLADOR	1	20209	
9	Prog	CAZA	1	5312	
10	Prog	RULETA	1	11418	

Presentación del programa «Listador»

y tiene una longitud de 16384 bytes o 16 Kbytes, por que

1 Kbyte = 1024 bytes

En caso particular, es la grabación de los datos de la zona de memoria correspon-

diente a la pantalla, ésta es la forma más usual de grabar las carátulas de los programas comerciales, recuerde la presentación de la cinta demostrativa del Spectrum 16 ó 48 K, "HORIZONTES".

Primero deberá dibujar en

pantalla e gráfico o texto que desee a base de "PLOT", "DRAW", "CIRCLE" o "PRINT". Cuando esté confeccionado se podrá grabar en cinta de la forma:

SAVE "MICRO" SCREEN\$

La palabra clave "SCREEN\$" es equivalente en este caso a

CODE 16384 6912

ya que la dirección de comienzo de la zona de pantalla es 16384 y tiene una longitud de 6912 bytes. Puede utilizar cualquiera de las dos fórmulas posibles

Para almacenar la zona de memoria correspondiente a los GDL puede utilizar, como

PROGRAMA 1

```

10 REM *****
*      LUKSO/BASIC      *
*      *****      *
*      LISTADOR      *
*      *****      *
20 REM [Comienzo Zona de 16384]
30 CLEAR 25555
40 BORDER 4: PAPER 4 INK 0 C
50 LET contador=1
60 RESTORE GO SUB 500
65 REM [Comienzo Zona de 16384]
70 INPUT "Nombre de la cinta"
80 IF LEN (s)>15 THEN LET (s)=(s)
90 FOR n=1 TO (15-LEN (s))
100 LET (s)=(s)+CHR$ 32
110 NEXT n
120 INPUT "Cara : ", LINE f$
130 IF LEN f$>1 THEN LET f$=f$
140 IF f$="" THEN LET f$=CHR$ 3
145 REM [Comienzo Zona de 16384]
150 LET d$=CHR$ 136+"CINTA "+CHR$ 32+
CHR$ 32+f$+CHR$ 32+CHR$ 32+CHR$ 32+
CHR$ 137
160 LET i$=""
170 LET d$=i$+d$
180 LET i$=""
190 LET s$=CHR$ 136+N Tipo NO
Nombre Comienzo Long "+CHR$ 133
200 LET d$=d$+s$+s$+s$+s$+s$
210 FOR n=2 TO 3
220 PRINT n,d$
230 NEXT n
235 REM [Comienzo Zona de 16384]
240 LET a$=""
250 LET j$=STR$ contador
260 IF LEN j$>2 THEN LET j$=CHR$
32+j$

```

```

270 RANDOMIZE USR 30000
280 FOR n=30001 TO 30030
290 IF PEEK n>31 AND PEEK n<144
THEN LET a$=a$+CHR$ PEEK n
300 IF PEEK n<32 OR PEEK n>143
THEN LET a$=a$+CHR$ 63
310 NEXT n
320 LET tipo=PEEK 30020
330 GO SUB 1000
340 LET w$=i$+CHR$ 32+i$+CHR$ 3
2+i$+CHR$ 32+c$+CHR$ 32+i$
350 PRINT w$ LPRINT w$
360 RANDOMIZE USR 30000
370 LET contador=contador+1
380 IF contador>15 THEN PRINT A
T 1,0,0+ RANDOMIZE USR 31999 PR
INT AT 19,0
410 IF contador>99 THEN LET con
tador=0
420 GO TO 240
499 STOP
500 REM [Comienzo Zona de 16384]
502 FOR n=30000 TO 30013
510 READ dato
520 POKE n,dato
530 NEXT n
540 RETURN
550 REM [Comienzo Zona de 16384]
560 DATA 62,0,221,33,68,117,17
570 DATA 17,0,00,603,00,0,200
1000 REM [Comienzo Zona de 16384]
1010 IF tipo=0 THEN LET i$="Prog"
1020 IF tipo=1 THEN LET i$="Data1N"
1030 IF tipo=2 THEN LET i$="Data$
1040 IF tipo=3 THEN LET i$="Byte"
1050 LET c$=STR$ IPEEK 30033+255
+PEEK 30034) LET c$="" +c$
LET c$=c$ (LEN (c$) TO LEN (c$)
1060 IF VAL (c$)>9999 AND tipo<3
THEN LET i$=""
1070 LET i$=STR$ IPEEK 30031+255
+PEEK 30032) LET i$="" +i$
LET i$=i$ (LEN (i$)-5) TO LEN (i$)
1080 RETURN

```



```

DIRECTORIO      M. MICROHOBBY.

      CODE A

NOMBRE MICROHOBBY

COMENTARIOS
Programas pertenecientes al nu-
mero 4 de MICROHOBBY/CASSETTE

PROPIETARIO R. P. R.

```

```

DIRECTORIO      M. MICROHOBBY.

      MENU DE OPCIONES

1 PRESENTACION.
2 DIRECTORIO.
3 EJECUCION.
4 INFORMACION.

```

```

DIRECTORIO      M. MICROHOBBY.

      INFORMACION

PROGRAMA UVE ATAC

COMENTARIOS
Una aventura de Cosme y los al-
ienigenas. Debe proteger el em-
balse de sus ataques.

PROGRAMADOR PAGO MARTIN

FECHA MAYO 85      PUNT 120

```

```

DIRECTORIO      M. MICROHOBBY.

      CARGA DE PROGRAMAS

PROGRAMA UVE ATAC

      MAYO 85

PULSE LA TECLA "PLAY" DEL CA-
SSETTE Y A CONTINUACION PULSE
LA TECLA "ENTER".

```

1	HECHIZO	11
2	HORMIGUERO	12
3	CARGADOR	13
4	OLYMPIA	14
5	SOLITARIO	15
6	UVE ATAC	16
7	CRETA	17
8	GUSANIN	18
9	SATURNO	19
10	GOMOKU	20

TECLEE EL NUM. DE PROGRAMA. >

Diversas pantallas del programa «Directorio».

ya se vio en el capítulo co-
respondiente

SAVE "nombre" CODE USA "a", 168

Grabación de matrices

Para la grabación de matri-
ces se utiliza el tipo "DATA"
Ejemplos:

SAVE "nombre" DATA X , J

— Almacena en cinta el
contenido de a matriz numé-
rica "X"

SAVE "nombre" DATA N\$ (J

— Almacena en cinta el
contenido de a matriz de ca-
dena "N\$".

VERIFY

PROGRAMA 2

```

10 REM *****
* CURSOR/BASIC *
* ***** *
* WILHELMIO *
* *****
20 GO SUB 4200
90 BORDER 1. PAPER 1. INK 7: C
L
100 REM PORTADA
105 GO SUB 5000
110 PRINT INVERSE 1, AT 3, 12, "CA
RA", PRINT " ", PRINT D$(21, 1
1 TO 13)
120 PRINT INVERSE 1, AT 8, 1, "NOM
BRE", PRINT " ", PRINT D$(21,
1 TO 10)
130 PRINT INVERSE 1, AT 11, 1, "CO
MENTARIOS", PRINT " ", PRINT
D$(21, 14 TO 63)
140 PRINT " ", D$(21, 34 TO 63)
150 PRINT " ", D$(21, 64 TO 93)
160 PRINT " ", D$(21, 94 TO 123)
170 PRINT INVERSE 1, AT 18, 1, "PR
OPIETARIO", PRINT " ", PRINT
D$(21, 14 TO 33)
180 PRINT "0", " PULSE 'ENTER
PARA CONTINUAR
190 PAUSE 0
200 LET a$=INKEY$
210 IF CODE a$ <> 13 THEN BEEP 0.
2, 15 GO TO 100
215 REM OPCIONES
220 GO SUB 5000
240 PRINT AT 3, 8, "MENÚ DE OPCIO
NES"
250 PRINT AT 4, 8, "
260 PRINT INVERSE 1, AT 6, 8, "1
PRINT " PRESENTACION."
270 PRINT INVERSE 1, AT 11, 8, "2
PRINT " DIRECTORIO."
280 PRINT INVERSE 1, AT 14, 8, "3"
PRINT " EJECUCION."
290 PRINT INVERSE 1, AT 17, 8, "4"
PRINT " INFORMACION."
295 PRINT AT 21, 1, "PULSE LA OPC
ION DESEADA."
300 REM SELECCION
305 PAUSE 0
308 PRINT AT 21 0, "
310 LET a$=INKEY$
320 IF a$="1" THEN PRINT FLASH
1, AT 6, 11, "PRESENTACION." PAUSE
75 GO SUB 5000 GO TO 110
330 IF a$="2" THEN PRINT FLASH
1, AT 11, 11, "DIRECTORIO." PAUSE
75 GO TO 1000
340 IF a$="3" THEN LET seleccion
r=3 PRINT FLASH 1, AT 14, 11, "EJE
CUCION." PAUSE 75 GO TO 2000
350 IF a$="4" THEN LET seleccion
r=4 PRINT FLASH 1, AT 17, 11, "INF
ORMACION." PAUSE 75 GO TO 2000
360 BEEP 0.2, 15 GO TO 295
1000 REM OPCION 1
1010 GO SUB 4000
1020 PRINT AT 21, 0, " PULSE 'ENT
ER" PARA RETORNAR."
1030 PAUSE 0
1040 LET a$=INKEY$
1050 IF CODE a$ <> 13 THEN BEEP 0.
2, 15 GO TO 1030
1060 GO TO 215
2000 REM OPCION 2
2010 GO SUB 4000
2020 GO SUB 4300
2030 LET flash=1
2050 GO SUB 4100
2060 PRINT AT 21, 1, "PULSE 'ENTE
R" O "DELETE"
2065 PAUSE 0
2070 LET a$=INKEY$
2080 IF a$ CHR$ 12 THEN BEEP 0.0
5, 20, LET flash=0 GO SUB 4100
GO TO 2020
2090 IF a$=CHR$ 13 THEN GO TO 31
10
2100 BEEP 0.2, 15 GO TO 2065

```

```

2110 CLS : GO SUB 5000
2120 IF seleccion=3 THEN GO TO 2
3000
2130 GO TO 3000
2500 REM OPCION 3
2510 PRINT AT 3, 7, "CARGA DE PROG
RAMAS"
2520 PRINT AT 4, 7, "
2530 PRINT INVERSE 1, AT 6, 1, "PRO
GRAMA", PRINT " ", D$(programa,
1 TO 10)
2540 PRINT INVERSE 1, AT 12, 1, "CO
NTADOR", PRINT " ", D$(programa,
11 TO 13)
2550 PRINT AT 16, 1, "PULSE LA TEC
LA 'PLAY' DEL CA
2560 PRINT AT 18, 1, "SETTE Y A C
ONTINUACION PULSE
2570 PRINT AT 20, 1, "LA TECLA "'E
NTER"
2580 PAUSE 0
2590 LET a$=INKEY$
2600 IF a$ <> CHR$ 13 THEN BEEP 0.
2, 15 GO TO 2580
2610 BEEP 0.05 20
2615 [
2620 LOAD D$(programa, 1 TO 10)
2699 STOP
3000 REM OPCION 4
3010 PRINT AT 2, 10, "INFORMACION"
3020 PRINT AT 3, 10, "
3030 PRINT INVERSE 1, AT 7, 1, "PRO
GRAMA", PRINT " ", D$(programa,
1 TO 10)
3035 PRINT INVERSE 1, AT 10, 1, "CO
MENTARIOS"
3040 PRINT AT 12, 1, D$(programa, 3
4 TO 63)
3050 PRINT AT 13, 1, D$(programa, 6
4 TO 93)
3060 PRINT AT 14, 1, D$(programa, 9
4 TO 123)
3070 PRINT INVERSE 1, AT 17, 1, "PR
OGRAMADOR.", PRINT " ", D$(progr
ama, 14 TO 25)
3080 PRINT INVERSE 1, AT 20, 1, "FE
CHA", PRINT " ", D$(programa, 26
TO 33)
3090 PRINT INVERSE 1, AT 20, 22, "C
ONT", PRINT " ", D$(programa, 11
TO 13)
3100 PRINT "0", " PULSE 'ENTER"
PARA RETORNAR.
3110 PAUSE 0
3120 LET a$=INKEY$
3130 IF a$ <> CHR$ 13 THEN BEEP 0.
2, 15 GO TO 3110
3140 CLS GO TO 220
3999 STOP
4000 REM SUB. CIF
4005 CIF : BEEP 0.05, 20 LET J=
1
4010 FOR I=1 TO 10
4020 IF I<10 THEN PRINT INVERSE
1, AT I+J, 2 I, PRINT " ", D$(I, 1
TO 10) PRINT INVERSE 1, AT I+J, 1
7, I+10, PRINT " ", D$(I+10, 1 TO
10)
4030 IF I=10 THEN PRINT INVERSE
1, AT I+J, 1 I, PRINT " ", D$(I, 1
TO 10) PRINT INVERSE 1, AT I+J, 1
7, I+10, PRINT " ", D$(I+10, 1 TO
10)
4035 LET J=J+1
4040 NEXT I
4050 RETURN
4100 REM SUB. FLASH
4120 IF programa=1 THEN LET x=4
LET y=(programa+2)-2: GO TO 41
40
4130 LET x=20 LET y=(programa-1
0)+2-2
4140 PRINT FLASH flash, AT y, x, D$
(programa, 1 TO 10)
4150 RETURN
4200 REM SUB. ULTIMO
4210 FOR I=1 TO 20
4220 IF C$(I, 1 TO 10)=
" THEN LET i:=i+1 GO TO 42
50
4230 LET ultimo=I

```

```

4430 LET programa=programa*10+VAL
L(a$),
4435 PAUSE 0
4440 IF programa>111110 THEN PRINT
AT 21,30,"", GO TO 4320
4450 RETURN
5000 REM [REDACTED]
5005 BEEP 0.05,20 CLS
5010 PRINT INVERSE 1," DIRECTORI
0 © MICROHARRY "
5020 RETURN
[REDACTED]
6010 LOAD "DIR" DATA D$(1)
6012 CLS
5020 PRINT FLASH 1,AT 10,10,"PAR
E LA CINTA."
6030 PRINT #0,"PULSE UNA TECLA P
ARA CONTINUAR." PAUSE 0
6040 GO TO 00

```

— VERIFY "T" DATA M\$ ()

Los campos "nombre" y "tipo" siguen las mismas reglas que en el caso de "SAVE".

a) Verificación de un programa BASIC.

VERIFY "nombre"

b) Verificación de un programa en C/M ó de bytes. Pueden utilizarse cualquiera de estas opciones

VERIFY "nombre" CODE comienzo longitud
VERIFY "nombre" CODE comienzo
VERIFY "nombre" CODE

Existen pequeñas diferencias en cuanto a la forma de verificar de cada una de ellas, siendo la más exacta la primera.

c) Verificación de una matriz numérica.

VERIFY "nombre" DATA extra ()

d) Verificación de una matriz de cadena

VERIFY "nombre" DATA letra\$ ()

Los programas almacenados como:

SAVE "nombre" SCREEN\$
0
SAVE "nombre" CODE 16384 8912

no pueden verificarse ya que al detectar el ordenador la cabecera de el programa, aparece en pantalla

Bytes: nombre

modificando por tanto la memoria de pantalla y dando lugar al correspondiente error.

LOAD

Acceso al teclado

VAL



MODOS K

VAL \$

Definición

La sentencia "LOAD" permite cargar en la memoria del ordenador un programa grabado en cinta de cassette

"LOAD" borra el programa existente en ese momento en el ordenador y sus variables.

La estructura general es:

SENTENCIA	ARGUMENTO
LOAD	"nombre" tipo

Ejemplo:

— LOAD "OH"
— LOAD "T" CODE
— LOAD "PP" DATA Z , ,
— LOAD "NT" DATA P\$ (, ,

Los campos "nombre" y "tipo" siguen las mismas reglas que en el caso de "SAVE".

a) Carga de programas BASIC

LOAD "nombre"

Si el programa fue almacenado con el tipo "LINE" automáticamente se autoejecutará

b) Carga de programas en C/M ó de bytes.

LOAD "nombre" CODE comienzo longitud
LOAD "nombre" CODE comienzo
LOAD "nombre" CODE

como casos específicos puede utilizarse para la carga de pantallas.

LOAD "nombre" SCREEN\$

La forma de cargar en pantalla una información es un tanto peculiar ya que su memoria está distribuida en tres zonas. Primero, se cargan los bytes cero de cada carácter (matriz de 8 por 8 pixel), de la primera zona, a continuación, los bytes uno y así sucesivamente hasta completar la primera zona. Las zonas dos y tres se cargan de la misma manera y al final se incluyen los atributos

El siguiente programa ayudará a comprender la secuencia de carga de la memoria de pantalla.

y para los GDU

```
10 PER *****
20 MEMORIA PANTALLA
30 *****
40 LET direccion=16384
50 LET longitud=8912
60 FOR h=direccion TO direccion+longitud
70 PUT h,255
80 NEXT h
```

LOAD nombre CODE ISK a

c) Carga de matrices numéricas.

LOAD "nombre" DATA extra ()

d) Carga de matrices de cadena

LOAD "nombre" DATA letra\$ ()

Al cargar una matriz, bien sea de cadena o numérica, no se borra el programa BASIC existente, pero si el contenido de cualquier matriz que anteriormente estuviera definida con el mismo nombre.

PROGRAMA 3

```

10 REM *****
  *      CURSO/BASIC      *
  * *****              *
  *      EDITOR          *
  * *****              *
12 BORDER & PAPER : IM# 0 : C
14 POKE 23656,0
20 DIM D$(21,12)
22 REM DATOS CARPULA
24 INPUT "NOMBRE DE LA CINTA"
  LINE A$
32 IF LEN A$>10 THEN LET A$=A$
  ( TO 10)
34 PRINT "NOMBRE" A$
36 LET D$(21,1 TO 10)=A$
40 INPUT "CARRA" LINE A$
42 IF LEN A$=1 THEN LET A$=A$
1) 44 PRINT "CARRA" > ",A$
46 LET D$(21,11)=A$
50 INPUT "PROPIETARIO" LINE
  A$
52 IF LEN A$>12 THEN LET A$=A$
  TO 12)
54 PRINT "PROPIETARIO" > ",A$
56 LET D$(21,14 TO 25)=A$
60 INPUT "COMENTARIOS" LINE
  A$
62 IF LEN A$>90 THEN LET A$=A$
  ( TO 90)
64 PRINT "COMENTARIOS" > A$
  LET D$(21,34 TO 123)=A$
70 PRINT #0, "PULSE UNA TECLA P
  PARA CONTINUAR." PAUSE 0
80 CLS
82 REM DATOS PROGRAMAS
100 FOR X=1 TO 20
110 INPUT "PROGRAMA" LINE A$
115 IF LEN A$>10 THEN LET A$=A$
  ( TO 10)
120 PRINT "PROGRAMA N."
122 IF X>9 THEN GO TO 126
124 PRINT
126 PRINT X, " > ",A$
130 LET D$(1 TO 10)=A$
140 INPUT "CONTADOR" LINE A$
145 IF LEN A$>3 THEN LET A$=A$
  TO 3)

```

```

150 PRINT "CONTADOR" > ",A
  $
160 LET D$(X,11 TO 13)=A$
170 INPUT "PROGRAMADOR" LINE
  A$
175 IF LEN A$>12 THEN LET A$=A$
  ( TO 12)
180 PRINT "PROGRAMADOR" > ",A
  $
190 LET D$(X,14 TO 25)=A$
200 INPUT "FECHA" LINE A$
205 IF LEN A$>8 THEN LET A$=A$
  TO 8)
210 PRINT "FECHA" > ",A
  $
220 LET D$(X,26 TO 33)=A$
230 INPUT "COMENTARIOS" LINE
  A$
235 IF LEN A$>90 THEN LET A$=A$
  ( TO 90)
240 PRINT "COMENTARIOS" > ",A
  $
250 LET D$(X,34 TO 123)=A$
260 POKE 23656,0
260 PRINT #0 "HAS PROGRAMAS 15/
  N)
262 LET A$=INKEY$
270 IF A$='S' THEN GO TO 300
280 IF A$='N' THEN GO TO 1000
290 GO TO 262
300 CLS
310 NEXT X
1000 PRINT "FIN DE EDICI
  ON"
1002 INPUT 0
1010 PRINT #0, "PULSE UNA TECLA P
  PARA CONTINUAR." PAUSE 0
1020 CLS
1022 REM GRABACION DE DATOS
1030 PRINT "INSERTE EN EL CASSET
  TE LA CINTA DONDE QUIERA GRAB
  AR LOS DATOS EDITADOS."
1040 PRINT #0 "PULSE UNA TECLA P
  PARA GRABAR." PAUSE 0
1045 CLS
1050 SAY "DIR DATA D$(X)"
1060 PRINT "GRABACION DE DATOS E
  FELI HOLA"
1070 PRINT #0, AT 0,0, "REBOBINE L
  A CINTA PARA VERIFICARY PULSE UN
  A TECLA." PAUSE 0
1080 CLS
1090 VERIFY "DIR" DATA D$(X)
1092 CLS
1100 PRINT "VERIFICACION DE DATO
  S CORRECTA"
1110 PAUSE 0

```

MERGE

Acceso al teclado

RND



MODOS E

MERGE

SYMBOL SHIFT

Definición

La sentencia "MERGE" permite combinar varios programas BASIC.

Su estructura general es

SENTENCIA	ARGUMENTO
MERGE	"nombre"

Ejemplo:

- MERGE "OK"

El argumento "nombre" sigue las mismas reglas que en el caso de "SAVE"

La secuencia de operación

es la siguiente

a) Cargar el primer programa de la forma acostumbrada.

LOAD "nombre"

b) Cargar el siguiente de la forma:

MERGE "nombre"

Si hubiera que combinar más programas, se procedería de la misma manera que en el punto "b".

NOTA

Hay que tener cuidado en que no haya líneas de programa con el mismo número, ya que el resultado podría ser desastroso.

Comodidad de uso

Las sentencias "LOAD", "VERIFY" y "MERGE" tienen cierta particularidad que les permita cargar, verificar o combinar el primer programa que encuentren, aunque no conozcamos su nombre; para ello debemos sustituir el nombre por una cadena vacía.

Ejemplos

```
LOAD ""  
VERIFY ""  
MERGE ""
```

Búsqueda de programas

Durante la búsqueda de programas para su carga, verificación o fusión con otros se van visualizando en pantalla todos aquellos programas o datos que el ordenador va encontrando. Esto puede servirnos de referencia para una mejor organización, ya que podemos utilizar el avance rápido hacia adelante o hacia atrás, de nuestro aparato de cassette, para ahorrar tiempo.

Referencias

Cuando el ordenador localiza un programa, se nos visualiza en pantalla su nom-

bre, así como una referencia indicativa del tipo de programa, éstas pueden ser de cuatro tipos:

a) Programas BASIC.

Programa número

b) Programas en C/M pantallas o bytes.

Bytes número

c) Matrices numéricas

Matriz número

d) Matrices de cadena.

Cadena número

Errores

En el manejo de estas sentencias pueden aparecer cualquiera de estos errores:

a) Fuera de memoria.

Fuera de memoria

Este error se produce cuando no hay suficiente memoria para cargar o combinar un programa, bytes o matrices.

b) Nombre de programa inválido.

Nombre inválido

Ocurre al grabar un programa con un nombre de más de diez caracteres o al asignarle una cadena vacía.

c) Error de carga en la cinta.

Se lepe la cinta

Aparece cuando el ordena-

dor no puede cargar un programa o cuando la verificación ha sido incorrecta.

Este error puede ser debido entre otras causas a:

- Cabeza lectora del cassette sucia.
- Pilas gastadas en exceso.
- Cintas apelmazadas.
- Cintas deterioradas.
- Etc.

Programas

El programa número "1" permite conocer el contenido de una cinta de cassette, dándonos información de los programas almacenados, así como una serie de datos útiles: su comienzo y su longitud. Si tenemos conectada una impresora los datos saldrán simultáneamente.

Una vez ejecutado, nos pregunta el nombre que queremos dar a la cinta (máx. 15 caracteres) y a continuación la cara (0). Posteriormente se pone el cassette, con la cinta, en marcha y a esperar que el ordenador visualice la información según vaya detectando las cabeceras de los programas.

El programa utiliza una pequeña rutina en código máquina, ya que parte de la información que suministra es inaccesible desde el BASIC.

El programa número "2" es un programa de utilidad que permite conocer el directorio de una cinta, es decir los programas que hay grabados, así como una serie de datos anexos: Nombre del programador, marcaje del contador del cassette, etc. También permite elegir un programa y cargarlo automáticamente.

Las instrucciones de manejo se encuentran en el propio programa

Los datos referentes a esa cinta deben ser editados con el programa número "3" que permite confeccionar la información de un máximo de 20 programas

Los programas deben ser grabados en el siguiente orden

- a) El programa número "2" como LINE 6000
- b) La tabla de datos que genera el programa número "3".
- c) A continuación los pro-

```
PROGRAMA N. 2    > QWERTY
CONTADOR         > 123
PROGRAMADOR      > D.F.G.
FECHA            > 23/04/85
COMENTARIOS      > ESTE PROGRAMA
EXPLORA EL TECLADO DEL SPECTRUM
```

Edición de datos con el "EDIT/DIR".

gramas correspondientes a esa cinta. Como el programa "directorio" es el primero de la cinta, para ejecutarlo rebo-

mina la cinta y teclea:

LOAD ""

GESTION DE IMPRESORA

Las impresoras son unos periféricos similares a una máquina de escribir, pero sin teclado incorporado, ya que las órdenes de escritura proceden del ordenador al que están conectadas.

Las utilidades de una impresora son diversas: desde una simple obtención en papel de los listados de programas hasta la confección, por ejemplo, de cartas con un sofisticado "PROCESADOR DE TEXTOS".

En este capítulo se van a explicar aquellas sentencias, que de un modo directo, gestionan las impresoras "ZX" o similares.

```
LPRINT
LIST
COPY
```

```
LPRINT
```

Acceso al teclado

LPRINT



MODO E

PAPER

Definición

La sentencia "LPRINT" realiza la misma tarea que

"PRINT" pero obteniendo los resultados por impresora en lugar de la pantalla.

Su estructura general es idéntica a la de "PRINT"; por tanto si tiene alguna duda consulte la página 71 y sucesivas.

Ejemplos:

- LPRINT "ORDENADOR"
- LPRINT a
- LPRINT 3/4 + 7
- LPRINT K\$

Con "LPRINT" pueden utilizarse las sentencias auxiliares "AT" y "TAB".

a) LPRINT AT

```
10 PRINT "ELEMENTOS DE LA TABLA"
20 LPRINT AT 10 "ORDENADOR"
40 PRINT
```

b) LPRINT TAB

```
10 PRINT "ELEMENTOS DE LA TABLA"
20 LPRINT TAB 10 "ORDENADOR"
40 PRINT
```

En general, el argumento de "LPRINT" no se imprime directamente, sino que se almacena en una memoria intermedia denominada "BUFFER DE IMPRESION DE LI-

NEA"; cuando este *buffer* se llena, la impresión se realiza.

Ejemplo

```
10 PRINT "ELEMENTOS DE LA TABLA"
20 LPRINT "ORDENADOR"
40 PRINT
```

A pesar de que en la línea 40 hay una impresión del elemento primero de la variable de cadena a\$, ésta no se realiza hasta que se completa el bucle Z (0 - 31), ya que 32 caracteres son los que se almacenan en la denominada memoria intermedia.

Hay otras ocasiones en las que se realiza la impresión sin estar lleno el buffer:

a) Cuando se utiliza "LPRINT" en comandos directos, independientemente de formato de impresión (coma o punto y coma).

Ejemplos

```
LPRINT "hoIn"
LPRINT "sepe"
PRINT "an"
LPRINT "an"
```

b) Dentro de un programa

— Cuando "LPRINT" no va acompañado de ningún formato

Ejemplo

PROGRAMA 1

```

10 REM *****
+ CLASO BASIC +
+*****+
+ IMPRESORA +
+*****+
20 LET n=0
20 LET primer=0
30 PRINT AT 0 0 INVERSE 1
TEST DE IMPRESORA
34 PRINT n0, Pulse una tecla

```

```

para comenzar
30 PAUSE 0 INPUT 0
40 FOR n=0 TO ultimo
50 LET caracter=primer+n
70 FOR x=caracter+longitud TO
caracter STEP 1
80 LPRINT CHR$ x
90 NEXT x
100 NEXT n
100 FOR z=0 TO 5
104 LPRINT
106 NEXT z
110 PRINT AT 21 0 INVERSE 1
FIN DEL TEST
120 PAUSE 0

```

```

10 LPRINT "uso
20 LPRINT "opu

```

- Cuando se requiera una nueva línea por alguno de los siguientes formatos:

COMA

```

10 LPRINT "Pulse una vez un tecla"
20 LPRINT "de 1 a 999"

```

APOSTROFE

```

10 LPRINT "Peras "Manzanas"

```

ELEMENTO "AT"

```

10 LPRINT AT 4 5 "AJA A"
20 LPRINT AT 10 0 "I, B RI"

```

Inserte un punto y coma al final de la línea 10 y observe la diferencia. Con la sentencia "AT" se ignora el indicativo de número de línea

ELEMENTO "TAB"

```

10 FOR n = 21 1, 0
20 LPRINT TAB n "a"
30 NEXT n

```

c) Al finalizar un programa si se ha quedado algo sin imprimir

```

10 LPRINT "a dos
20 PAUSE 0

```

Al pulsar cualquier tecla, se imprime la cadena "ad os".

```

@ / . , - , + * ) ( % % $ # '
P@?>=, 9876543210/.- , + * ) ( % % $ # '
BA@?>=, 9876543210/.- , + * ) ( % % $ # '
CB@?>=, 9876543210/.- , + * ) ( % % $ # '
DCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
EDCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
FEDCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
GFEDCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
HGFE DCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
IHGFEDCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
JIHGFEDCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
KJIHGFEDCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
LKJIHGFEDCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
MLKJIHGFEDCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
NMLKJIHGFEDCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
ONMLKJIHGFEDCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
PONMLKJIHGFEDCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
QPONMLKJIHGFEDCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
RQPONMLKJIHGFEDCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
SRQPONMLKJIHGFEDCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
TSRQPONMLKJIHGFEDCBA@?>=, 9876543210/.- , + * ) ( % % $ # '
UTSRQPONMLKJIHGFEDCBA@?>=, 9876543210/.- , + * ) ( % % $ # '

```

Ejemplo del programa "Test"

que se encontraba almacenada en el buffer de impresión

LIST

"LIST" en impresora
Su estructura general es

SENTENCIA	ARGUMENTO
LIST	n de línea

Ejemplos

- LLIST
- LLIST 30
- LLIST a
- LLIST 5 * b

Cuando el número de línea se omite, el ordenador interpreta que el listado es a partir de a primera línea

A diferencia de "LIST" el listado se imprime todo seguido, ya que no aparece el conocido mensaje

Acceso al teclado

LLIST



MODO E

FLASH

Definición

"LLIST" es el equivalente a

scrn1

Los listados por impresora son muy convenientes, ya que ayudan en la depuración de programas a poder localizar con mayor facilidad las sentencias.

COPY

Acceso al teclado

LN



MODO K

BEEP

Definición

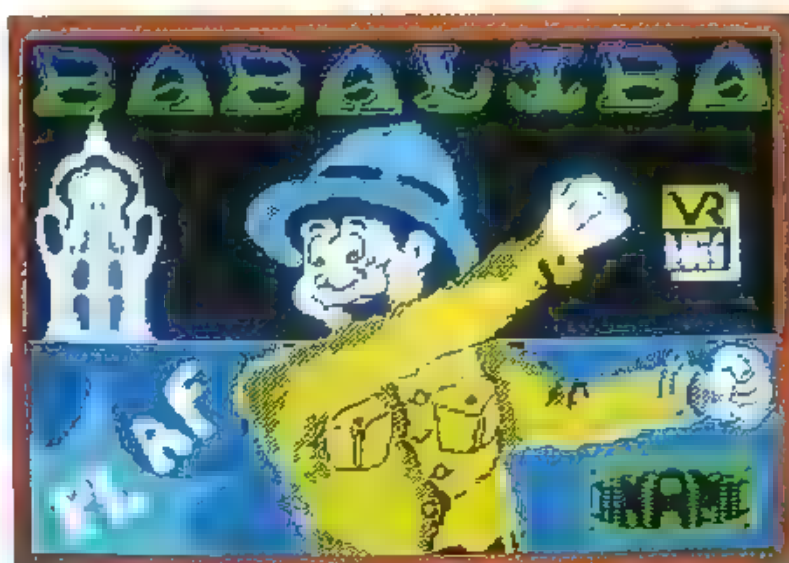
La sentencia "COPY" permite sacar por impresora una copia de la imagen visualizada en pantalla.

"COPY" no necesita de ningún argumento para ejecutarse.

La reproducción que hace "COPY" de la pantalla se basa en la impresión de los pixel con color de "tinta", por tanto, no se sorprenda si al intentar realizar un "COPY" del dibujo proporcionado por el programa de la página 227, no se imprime absolutamente nada, ya que está realizado a base de bloques de color "papel" con el carácter "espacio"; si desea imprimirlo, modifique la siguiente línea.

1030 PR NT NK color AT y x, █

Tampoco se imprime nada si intenta realizar un "COPY" de los listados que proporciona el ordenador al pulsar la



Sólo los pixel con color de tinta se imprimen al realizar un "copy".

tecla "ENTER", ya que éstos se borran al ejecutarse una sentencia o comando.

Realice prácticas con los ejercicios de las sentencias "PLOT", "DRAW" y "CIRCLE"

Programa de utilidad

El programa n.º 1 permite realizar uno de los típicos TEST de impresora, en los

que los caracteres se van desplazando hacia la derecha; de esta manera se puede, mediante una rápida ojeada, comprobar el correcto funcionamiento de una impresora

Otras impresoras

En el mercado existen diversos *Interfaces* para poder

Letra Normal de matriz de puntos
Letra Itálica o cursiva
 Letra de calidad (NLQ)
 Escritura en negrita
Escritura subrayada
 Paso de escritura condensado
 Paso de escritura "Elite"
 Paso de escritura "Pica"
Itálica condensada
 Línea con superíndices y subíndices
 La letra más pequeña posible
 Normal expandido
Itálica expandida
 Condensado expandido
Itálica condensada expandida

Diversos tipos de letra, proporcionados por una impresora de calidad

acoplar cualquier impresora, que no sea "ZX", al Spectrum.

Los protocolos de comunicación más importantes, por los que se rigen estas impresoras, son:

CENTRONICS
RS 232

El protocolo "CENTRONICS" se basa en el envío paralelo de la información, mientras que la "RS 232" lo realiza en serie.

La mayoría de estos interfaces ocupan una pequeña parte de la memoria de Spectrum y necesitan de un Software que los gestione.

Tipos de Impresora

En la actualidad existe una gran variedad. Atendiendo a

su sistema de funcionamiento se pueden clasificar de la siguiente manera:

MARGARITA
 AGUJA
 BUBBLES
 LINEAS
 INYECCION DE TONER
 ELECTROSTATICAS
 LASER
 ETC.

Las impresoras de Margarita tienen la ventaja de tener un tipo de letra idéntico al proporcionado por una máquina de escribir, pero tienen el inconveniente de que su velocidad expresada en "cps" (caracteres por segundo) es relativamente baja.

Las impresoras de agujas son las más utilizadas en los

ordenadores personales, ya que su relación servicio/precio es bastante adecuada.

La velocidad de estas impresoras es superior a las de Margarita, pero su tipo de letra es más imperfecto, ya que las realiza a base de puntos.

Las impresoras térmicas también se utilizan en los ordenadores personales; en éstas, el tipo de papel utilizado es especial, ya que cambian de color al calentarse.

El resto de impresoras no son de uso frecuente con ordenadores personales, ya que su precio es bastante elevado y están diseñadas para equipos donde el volumen de impresión es grande, por ejemplo, las de líneas imprimen una línea de una sola vez y las de láser hasta una página.

```

FIRMWARE ID: 21066 REV A
FIRMWARE ID: 21065 REV A
GENERATOR INTERNAL
1K RAM

```

```

FORM LENGTH          12 INCH
BOTTOM OF FORM SKIP  YES
LINES PER INCH       6
CHARACTER PER INCH   10
CR = CR LF           YES
LF AT LINE END       NO
CHARACTER SET         US ASCII
SLASHED ZERO         YES
TEST MODUS           ROLLING ASCII
                     END OF MENU.

```

```

CHANGE MENU          YES
CHANGE FORM LENGTH
    4 INCH            NO
    5 INCH            NO
    6 INCH            NO
    8 INCH            NO
    8.5 INCH          NO
    11 INCH           NO
    12 INCH           NO
    14 INCH           NO CHANGE

```

```

BOTTOM OF FORMAT SKIP YES
CHANGE PRINT FORMAT   YES
CHANGE LPI
    4                 NO
    8                 NO CHANGE
CHANGE CPI
    10                NO
    12.5              NO
    16 2/3            NO
    20                NO CHANGE

```

```

CR = CR LF           YES
LF AT LINE END       NO
CHANGE CHARACTER SET  YES
US ASCII              NO
UK ASCII              NO
FRENCH/BELGIAN       NO
GERMAN                NO
ITALIAN               NO
SWEDISH/FINNISH       NO
DANISH/NORWEGIAN     NO
SPANISH               YES
SLASHED ZERO         YES
CHANGE TEST MODUS    YES
ROLLING ASCII        NO
GENERATOR TEST       YES
                     END OF MENU.

```

Menú de opciones de Impresión programables por el usuario.

280 MICROBASIC

Elección de una impresora

A la hora de elegir una impresora deben tenerse en cuenta ciertos factores que intervendrán directamente en su precio.

- La velocidad de impresión en cps

- Si imprime en dos sentidos (bidireccional).

- Número de caracteres por línea

- Tipo de papel

- Rollo
- Perforado
- Térmico
- Hojas

- Tipo de tinta

- Juegos de caracteres que incorpora

- Si tiene un sistema de Fricción y de Arrastre combinado que permite introducir o bien hojas sueltas o bien papel perforado.

- Si imprime en varios colores.

La elección deberá ir en concordancia con el tipo de utilidad a la que se vaya a destinar y con el volumen de trabajo que deberá soportar.

Caracteres de control

La mayoría de las impresoras utilizan algunos caracteres ASCII para cambiar el tamaño de las letras, la distancia entre líneas, el tipo de letra, si deben ir subrayadas o en negrita, etc

Estos códigos dependen del tipo de impresora, por tanto deberá leer las instrucciones específicas de la que esté manejando; pero en general la forma de introducir estos comandos es:

```
PRINT CHR$ n
```

donde "n" es el código de control a enviar.

US ASCII

'###&'()*+,-./0123456789:;<=?
@ABCDEFGHIJKLMN[OPQRSTUVWXYZ\]^_`abcdefghijklmnopqrstuvwxyz{|}~
'###&'()*+,-./0123456789:;<=?
@ABCDEFGHIJKLMN[OPQRS[TUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~

UK ASCII

'###&'()*+,-./0123456789:;<=?
@ABCDEFGHIJKLMN[OPQRSTUvwXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}>
'###&'()*+,-./0123456789:;<=?
@ABCDEFGHIJKLMN[OPQRS[TUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}>

FRENCH/BELGIAN

'###&'()*+,-./0123456789:;<=?
@ABCDEFGHIJKLMN[OPQRS UVWXYZ^`_`abcdefghijklmnopqrstuvwxyzéè
'###&'()*+,-./0123456789:;<=?
@ABCDEFGHIJKLMN[OPQRS[TUVWXYZ^`_`abcdefghijklmnopqrstuvwxyzéè

GERMAN

'###&'()*+,-./0123456789:;<=?
@ABCDEFGHIJKLMN[OPRS TUvwXYZaöü_`abcdefghijklmnopqrstuvwxyzäüß
'###&'()*+,-./0123456789:;<=?
@ABCDEFGHIJKLMN[OPRS[TUvwXYZaöü_`abcdefghijklmnopqrstuvwxyzäüß

ITALIAN

'###&'()*+,-./0123456789:;<=?
@ABCDEFGHIJKLMN[OPQRS UVwxyz^`_`abcdefghijklmnopqrstuvwxyzàéè
'###&'()*+,-./0123456789:;<=?
@ABCDEFGHIJKLMN[OPRS[TUVwxyz^`_`abcdefghijklmnopqrstuvwxyzàéè.

SWEDISH/FINNISH

'###&'()*+,-./0123456789:;<=?
@ABCDEFGHIJKLMN[OPQRS UVwxyz^`_`abcdefghijklmnopqrstuvwxyzäöå
'###&'()*+,-./0123456789:;<=?
@ABCDEFGHIJKLMN[OPQRSTUvwxyz^`_`abcdefghijklmnopqrstuvwxyzäöå

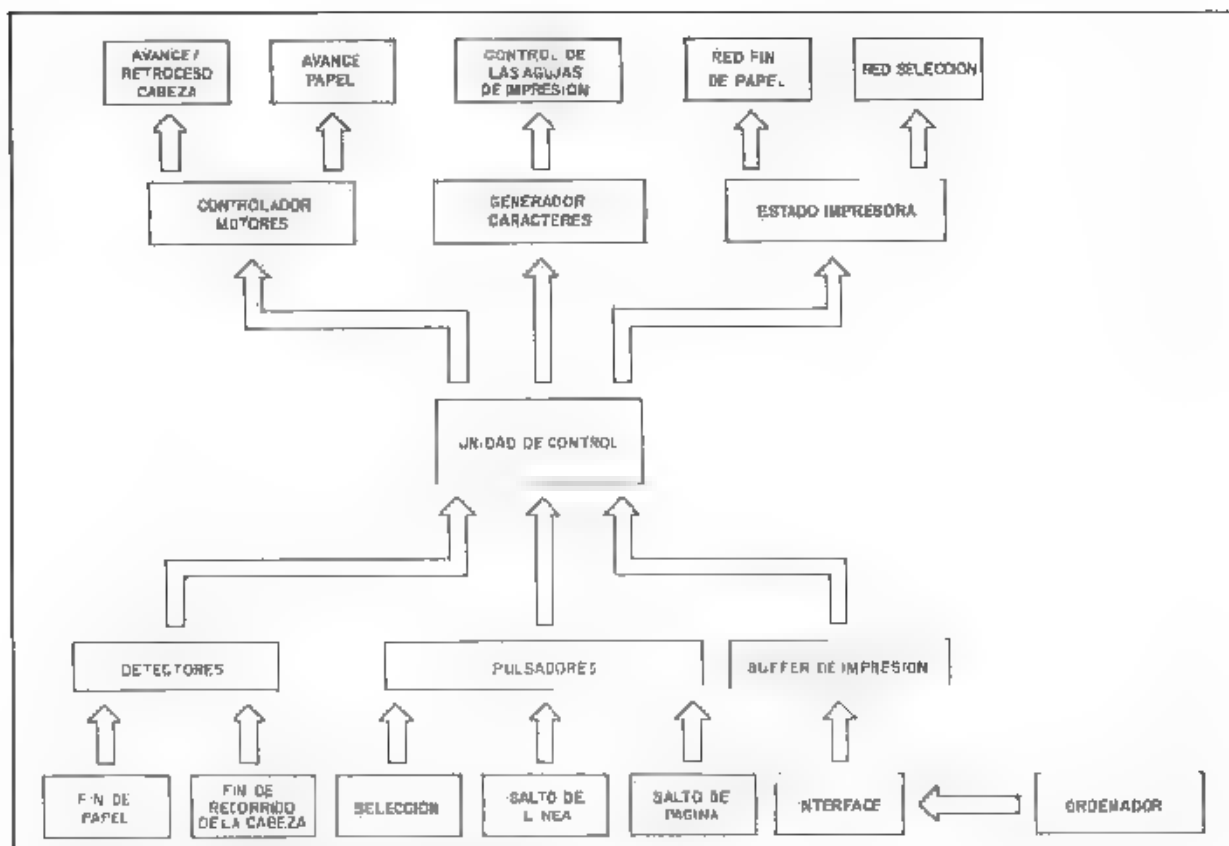
DANISH/NORWEGIAN

'###&'()*+,-./0123456789:;<=?
@ABCDEFGHIJKLMN[OPQRSTUvwXYZ@A@_`abcdefghijklmnopqrstuvwxyz'æø
'###&'()*+,-./0123456789:;<=?
@ABCDEFGHIJKLMN[OPQRS[TUVWXYZ@A@_`abcdefghijklmnopqrstuvwxyz'æø

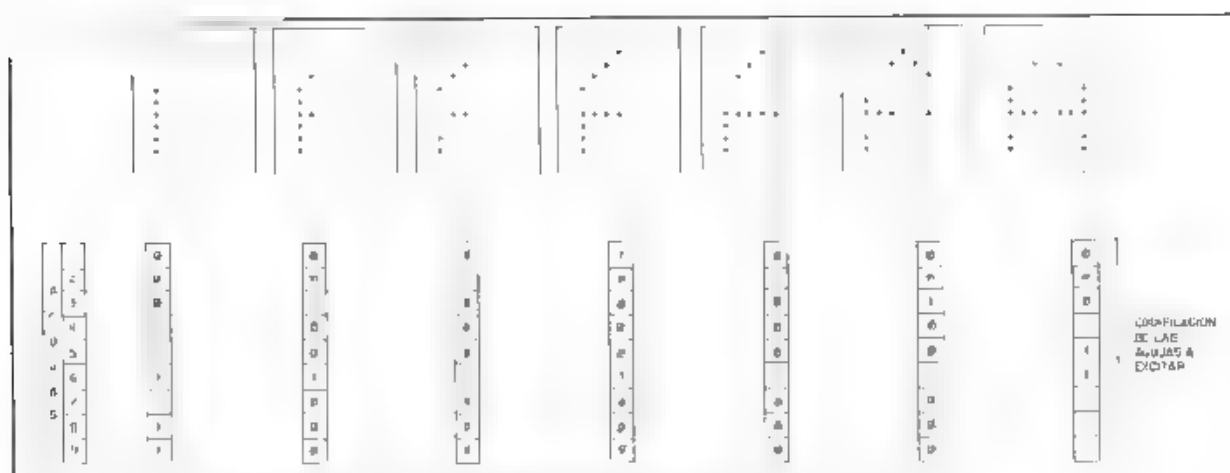
SPANISH

'###&'()*+,-./0123456789:;<=?
@ABCDEFGHIJKLMN[OPQRSTUVWXYZiñü_`abcdefghijklmnopqrstuvwxyzíñó
'###&'()*+,-./0123456789:;<=?
@ABCDEFGHIJKLMN[OPQRSTUvwXYZiñü_`abcdefghijklmnopqrstuvwxyzíñó

ACTUAL CHARACTERSET SPANISH



Esquema de bloques de una impresora.



Formación del carácter "A" con una impresora de agujas de 9 x 7.

GESTION DE IMPRESORAS -

Las impresoras son unos periféricos similares a una máquina de escribir, pero sin teclado incorporado, ya que las órdenes de escritura proceden del ordenador al que estén conectados.

Las utilidades de una impresora son ó versas, desde la simple obtención en papel de los listados de programas hasta la conexión por ejemplo, de cartas con un sofisticado "PROCESADOR DE TEXTOS".

Las impresoras son una herramienta imprescindible en un "Procesador de Textos"

INTERFACE-1

El "Interface-1" es un controlador de periféricos que permite manejar con el empleo de ciertas sentencias:

- Hasta un máximo de ocho Microdrives (unidades de almacenamiento)

- La red de área local mediante la cual se pueden comunicar hasta 64 Spectrum

- El Interface RS-232, para conectar aquellos periféricos que utilicen este protocolo de transmisión de datos en serie.

Prolonga el conector de expansión, de manera que se puedan añadir más periféricos al Spectrum

Canales y Corrientes

El intercambio de datos entre el Spectrum y sus periféricos se realiza a través de los denominados *canales de comunicación*, (ver pag. 75).

Para identificar una comunicación debe especificarse el canal y la corriente

Los canales pueden ser de entrada, salida o de entrada/salida

ENTRADA

Teclado

SALIDA

- Pantalla del televisor o monitor

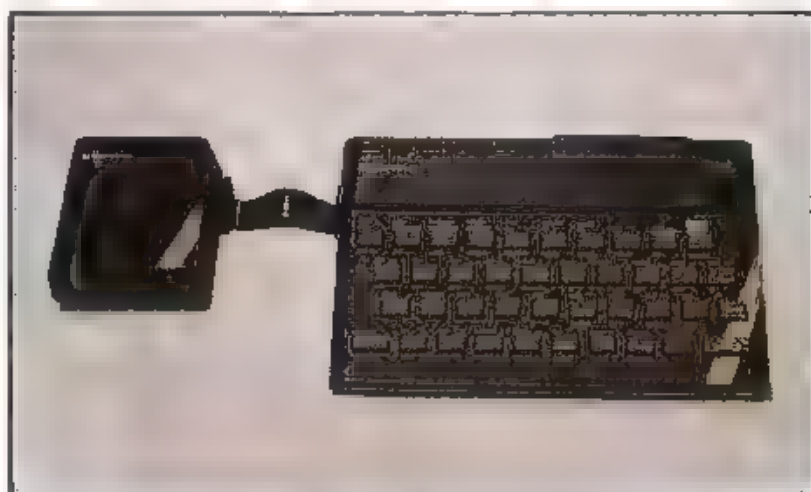
- Impresora.

ENTRADA/SALIDA

Fichero Microdrive.

- Un Spectrum conectado a la red.

- El interface RS-232.



Conexión del Microdrive.

Los distintos indicativos del canal están especificados en la figura adjunta

El flujo de datos puede llegar a recibirse, de estos canales, por varios caminos o corrientes; en total el Spectrum dispone de 16, numerados del #0 al #15

Las corrientes #0 a #3 tienen una asignación definida por el sistema operativo; las restantes #4 a #15 están libres para poder ser definidas por el propio usuario.

Asignación de canales y corrientes

Para poder asignar un "canal" a una «corriente» debe utilizarse la sentencia "OPEN #"

OPEN #

Acceso al teclado

GREEN
INV. VIDEO

MODO E



SYMBOL SHIFT

OPEN #

Su estructura general es:

SENTENCIA	ARGUMENTO
OPEN #	corriente, canal

Ejemplos

a) Asignación de la corriente #6 a un fichero de la unidad "3" Microdrive denominado "curso".

OPEN # 6 "m" 3 "curso"

b) Asignación de la corriente #8 a la estación 4 de la red de área local.

OPEN # 8 "r", 4

c) Asignación de la corriente # 15 a una impresora conectada al interface RS-232.

OPEN # 15 "r"

d) Para asignar una corriente a cualquiera de los canales: "k" (teclado) "s" (pantalla) o "p" (impresora) debe utilizarse necesariamente el separador "coma"; en los demás canales, puede utilizarse indistintamente "coma" o "punto y coma"

OPEN # 5 "s"

OPEN # 7 "p"

OPEN # 0, "k"

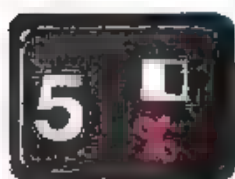
Desactivación de canales y corrientes

Para quitar la asignación de un "canal" a su correspondiente "corriente", se utiliza la sentencia "CLOSE #"

CLOSE #

Acceso al teclado

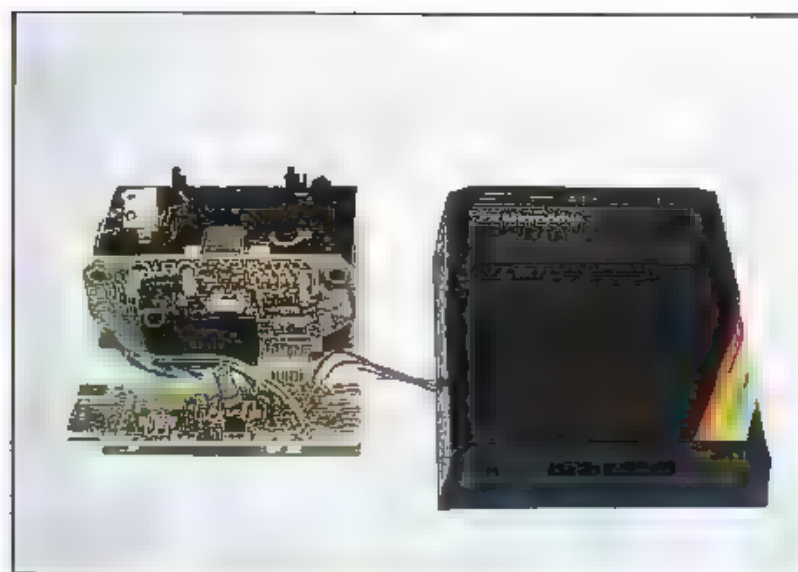
CYAN



MODO E

BOL SHIFT

CLOSE #



Interior del Microdrive

Su estructura general es.

SENTENCIA	ARGUMENTO
CLOSE #	corriente

Ejemplos:

- CLOSE # 6
- CLOSE # n
- CLOSE # Z + 1
- CLOSE # VAL a\$

Al realizar la desactivación, los datos que estuvieran almacenados en la memoria intermedia serán transmitidos a su correspondiente canal.

El Microdrive

Es un periférico diseñado para almacenar gran cantidad de datos, con un tiempo de acceso de lectura/escritura superior al proporcionado por un cassette. La información se almacena en unos *cartuchos* intercambiables.

Las operaciones que pueden realizarse con el Microdrive son:

- Formatear o inicializar cartuchos.
- Almacenar y verificar

programas en cartucho

- Cargar programas
- Borrar *ficheros* de datos y programas.
- Combinar los programas residentes en memoria con los archivados en cartucho
- Catalogar cartuchos, es decir, obtener un *directorio* o listado de los programas grabados en él.
- Abrir y cerrar *ficheros* de datos.
- Grabar y leer datos de un *fichero*.
- Mover datos entre *ficheros*.

FORMAT

Acceso al teclado

BLACK

DELETE

MODOS E



BOL SHIFT

FORMAT

PROGRAMA 1

```

10 REM *****
* CURSOR/BASIC *
* *****
* FICHERO *
* *****

20 BORDER 4: PAPER 4: INK 0 C
LS
22 POKE 23658,8
30 LET Ultimo=0
40 LET e$="" LET f$=""
100 DIM a$(100,40)
500 REM MENU
510 CLS CLOSE #4
520 PRINT INVERSE 1, " FICHE
RO BIBLIOTECARIO
522 PRINT INVERSE 1, AT 5,10, "
OPCIONES
530 PRINT AT 9,10, "1 GENERAR"
540 PRINT AT 12,10, "2 ALMACENAR
550 PRINT AT 15,10, "3 RECUPERAR
552 PRINT AT 18,10, "4 VISUALIZA
R
560 PRINT #0; AT 1,4, "Pulse la o
pcion deseada"
562 PAUSE 0
570 LET c$=INKEY$
580 IF c$="1" OR c$="4" THEN BE
EP 0.2, -10: GO TO 562
590 BEEP 0.2,10 GO TO (VAL c$)
*1000
1000 REM GENERAR
1002 IF Ultimo=0 THEN GO TO 1020
1004 CLS
1006 IF Ultimo>0 AND Ultimo<=99
THEN GO TO 1015
1010 IF Ultimo>99 THEN PRINT INV
ERSE 1, AT 4,5, " FICHERO COMPLET
O "; INVERSE 0, AT 10,7: 3 NUEVO
FICHERO", AT 15,7: 4 MENJ OPCIONE
S"; #1 AT 1,4, "Pulse la opción de
seada"
1011 PAUSE 0 LET z$=INKEY$
1012 IF z$="N" THEN BEEP 0.2,10
DIM a$(100,40) LET Ultimo=0 L
ET e$="" LET f$="" GO TO 1020
1013 IF z$="N" THEN BEEP 0.2,10
GO TO 500
1014 BEEP 0.2, -10: GO TO 1011
1016 PRINT AT 7,6, "0 NUEVO FICHE
RO", AT 14,8, "1 CONTINUAR" #1, AT
1,4, "Pulse la opción deseada"
1016 PAUSE 0 LET z$=INKEY$
1017 IF z$="N" THEN BEEP 0.2,10
DIM a$(100,40) LET Ultimo=0 L
ET e$="" LET f$="" GO TO 1020
1018 IF z$="0" THEN BEEP 0.2,10
CLS GO TO 1020
1019 BEEP 0.2, -10: GO TO 1015
1020 CLS PRINT INVERSE 1, "
GENERAR FICHA
1030 PRINT AT 7,0, "1 WHERE"
1040 PRINT AT 14,0, "2 AUTOR"
1050 FOR n=Ultimo+1 TO 100
1052 LET a$(n,1)=STR$(n)
1054 PRINT AT 0,22, INVERSE 1;n$
(1 TO 3)
1060 INPUT "Nombre> "; LINE b$
BEEP 0.2,10
1062 IF b$="FIN" THEN GO TO 500
1064 IF b$="" THEN GO TO 1060
1070 IF LEN b$>20 THEN LET b$=b$
(1 TO 20)
1080 PRINT AT 7,9, b$
1090 LET a$(n,1 TO 20)=b$
1100 INPUT "Autor> "; LINE c$, B
EEP 0.2,10
1102 IF c$="" THEN GO TO 1100
1110 IF LEN c$>20 THEN LET c$=c$
(1 TO 20)
1120 PRINT AT 14,9, c$
1130 LET a$(n,21 TO 40)=c$
1140 LET Ultimo=Ultimo+1

```

```

1150 PAUSE 50
1180 PRINT AT 7,9, "
1170 PRINT AT 14,9, "
1180 NEXT n
1190 GO TO 500
2000 REM ALMACENAR
2010 CLS
2020 PRINT INVERSE 1, " GRAB
ACION DE FICHERO
2030 PRINT AT 5,0, "1 WHERE", AT
10,0, "2 FECHA", AT 15,0, "3 CLAVE
"
2040 INPUT "Nombre> "; LINE b$
BEEP 0.2,10
2050 IF b$="" THEN GO TO 2040
2060 IF LEN b$>20 THEN LET b$=b$
(1 TO 20)
2070 PRINT AT 5,9, b$
2080 INPUT "Fecha> "; LINE c$, B
EEP 0.2,10
2090 IF c$="" THEN GO TO 2080
2100 IF LEN c$>20 THEN LET c$=c$
(1 TO 20)
2110 PRINT AT 10,9, c$
2112 LET d$=""
2120 FOR n=0 TO 9
2130 PAUSE 0
2140 IF INKEY$=CHR$(13) THEN BEEP
0.05,10 GO TO 2180
2150 LET d$=c$+INKEY$
2160 PRINT AT 15,9+n, "z"
2162 BEEP 0.05,10
2170 NEXT n
2180 PRINT AT 21,0, INVERSE 1, "
INTRODUZCA EL CART. MICRODRIVE"
PAUSE 100
2190 PRINT #1, AT 1,1, "Pulse una
tecla para comenzar" PAUSE 0 I
NPUT 0 BEEP 0.2,10
2200 PRINT AT 21,0, INVERSE 1; "
GRABANDO CARTUCHO MICRODRIVE"
2210 OPEN #4, "a", 1, b$+"CLU"
2220 PRINT #4, b$+c$+d$+Ultimo+1
2230 CLOSE #4
2240 OPEN #4, "a", 1, b$+"DAT"
2250 FOR n=1 TO Ultimo
2260 PRINT #4 a$(n)
2262 NEXT n
2270 CLOSE #4
2272 BEEP 0.2,10
2280 PRINT AT 21,0, INVERSE 1, "
GRABACION EFECTUADA
2290 PRINT #1 AT 1,1, "Pulse una
tecla para retornar"
2300 PAUSE 0: BEEP 0.2,10: GO TO
500
3000 REM RECUPERAR
3010 CLS
3020 PRINT INVERSE 1; " LEGT
URA DE FICHERO
3030 PRINT AT 5,0, "1 WHERE", AT
10,0, "2 FECHA"
3040 INPUT "Nombre> "; LINE b$
BEEP 0.2,10
3050 IF b$="" THEN GO TO 3040
3060 IF LEN b$>20 THEN LET b$=b$
(1 TO 20)
3070 PRINT AT 5,9, b$
3072 LET d$=""
3080 FOR n=0 TO 9
3090 PAUSE 0
3092 IF INKEY$=CHR$(13) THEN BEEP
0.2,10 GO TO 3140
3100 LET d$=d$+INKEY$
3110 PRINT AT 10,9+n, "z"
3120 BEEP 0.05,10
3130 NEXT n
3140 PRINT AT 21,0, INVERSE 1, "
INTRODUZCA EL CART. MICRODRIVE"
PAUSE 100
3150 PRINT #1, AT 1,1, "Pulse una
tecla para comenzar" PAUSE 0 I
NPUT 0 BEEP 0.2,10
3160 PRINT AT 21,0, INVERSE 1, "
CARGANDO CARTUCHO MICRODRIVE"
3170 OPEN #4, "a", 1, b$+"CLU"

```

```

3100 INPUT #4, LINE #$: LINE #$,
LINE #$,ultimo
3100 IF #<># THEN NEU
3102 LET #:= LET #:=
3200 CLOSE #4
3202 DIM a$(100,40)
3210 OPEN #4,"a",1,b$+".DAT"
3220 FOR n=1 TO ultimo
3230 INPUT #4, LINE #$
3240 LET a$(n,)=#
3242 NEXT n
3244 CLOSE #4
3260 PRINT AT 21,0; INVERSE 1;";
FICHERO CARGADO
3262 BEEP 0.2,10
3270 PRINT #1,AT 1,1,"Pulse Una
tecla para retornar"
3280 PAUSE 0 BEEP 0.2,10. GO TO
500
4000 REN :SUMA:RA
4002 CLS : PRINT INVERSE 1;";
VISUALIZACION DE FICHERO
4004 PRINT AT 10,10,"PANTALLA
AT 15,10,"IMPRESORA",#1,AT 1,
4,"Pulse la opción deseada"
4006 PAUSE 0

```

```

4008 IF INKEY$="P" THEN BEEP 0.2
,10: CLS : LET periferico=2 GO
TO 4020
4010 IF INKEY$="I" THEN BEEP 0.2
,10: CLS : LET periferico=3 GO
TO 4014
4012 BEEP 0.2,-10, GO TO 4006
4014 PRINT AT 21,0; INVERSE 1;";
CONECTE LA IMPRESORA
4016 PAUSE 100 PRINT #1," Pulse
una tecla para continuar" PAUS
E 0 BEEP 0.2,10 CLS
4020 PRINT #periferico;" FICHERO
" #";"FECHA";"RA"
4022 FOR n=1 TO ultimo
4030 PRINT #periferico, INVERSE
1;n; INVERSE 0,TAB 4,a$(n,1 TO 2
0) TAB 4,a$(n,21 TO 40)
4040 NEXT n
4042 IF periferico=3 THEN FOR n=
1 TO 15 LPRINT : NEXT n
4044 BEEP 0.2,10
4050 PRINT #1,AT 1,1. Pulse una
tecla para retornar"
4060 PAUSE 0 BEEP 0.2,10. GO TO
500

```

Definición

La sentencia "FORMAT" permite entre otras cosas, formatear o inicializar un cartucho.

Su estructura general es

SENTENCIA	ARGUMENTO
FORMAT	"m"; unidad; "nombre"

Ejemplos:

- FORMAT "m"; 1; "BAS"
- FORMAT "m"; 5; "uon"
- FORMAT "m"; 3; "Aster"
- FORMAT "m"; 8; "BUG"

El indicativo "m" hace referencia al canal seleccionado (Microdrive)

El parámetro *unidad* es un numero entero comprendido entre 1 y 8, e indica la unidad de Microdrive que deseamos seleccionar.

El parámetro "nombre" es una cadena formada por un máximo de 10 caracteres, e indica al nombre asignado al cartucho, este se visualiza al pedir el *directorio* del cartucho.

El formateado de un cartucho "virgen" debe ser realizado antes de su utilización.

Mientras se realiza la inicialización de un cartucho, el Sistema Operativo chequea las zonas donde no puede LEER o ESCRIBIR, marcándolas, para tenerlas en cuenta en operaciones de acceso posteriores.

ADVERTENCIA

- Al formatear un cartucho con información, ésta se destruye.

- No debe sacarse un cartucho cuando el "LED" indicativo de acceso se encuentra encendido.

- No debe desconectarse el Spectrum con un cartucho insertado en una unidad Microdrive.

En los dos últimos casos deberá procederse a una nueva inicialización del cartucho.

La capacidad de almacenamiento libre de un cartucho formateado es superior a 85

Kbytes, pudiendo llegar hasta 100.

100 Kbytes = 102400 bytes

CAT

Acceso al teclado

GRAPHICS



MODO E

CAT

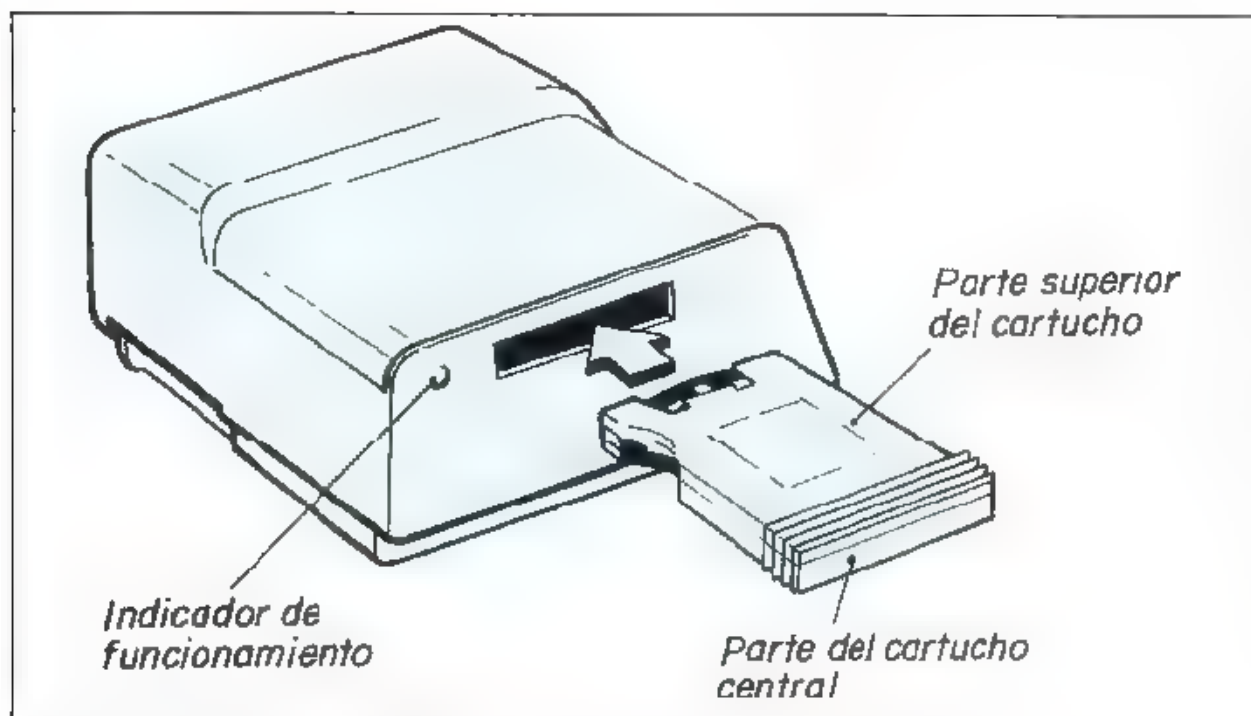
SYMBOL SHIFT

Definición

Permite obtener una lista o catálogo de todos los ficheros almacenados en un cartucho.

Su estructura general es:

SENTENCIA	ARGUMENTO
CAT	unidad



Unidad de Microdrive mostrando la correcta inserción del cartucho.

Ejemplos

CAT 5

- CAT 1
- CAT (n - 1) * 2
- CAT VAL b\$

La forma de presentar un catálogo o directorio es la siguiente:

- Nombre asignado al formatear

- Lista ordenada por orden alfabético de todos los ficheros.

- Capacidad libre expresada en Kbytes.

Utilizando otra estructura se puede enviar el catálogo a otro canal de comunicación.

SENTENCIA	ARGUMENTO
CAT	# corriente, unidad

Ejemplo.

CAT # 3 5

La anterior instrucción envía el catálogo de la unidad Microdrive número 5 a una impresora del tipo ZX

Grabación y carga

Todas las opciones de grabación y carga de programas en cassettes están disponibles para su utilización con el Interface-1, solamente varía la sintaxis de la instrucción.

GRABACION

SENTENCIA	ARGUMENTO
SAVE	* "n" unidad "nombre"

Ejemplos:

- SAVE * "m" 2; "cuat"
- SAVE * "m"; 3; "pelo"

El símbolo del asterisco indica al sistema operativo que la grabación debe efectuarse a través del Interface-1, y no del cassette.

VERIFICACION

SENTENCIA	ARGUMENTO
VERIFY	* "n" unidad "nombre"

Ejemplos:

- VERIFY * "m"; 8; "vale"
- VERIFY * "m"; 1; "OKEY"

CARGA

SENTENCIA	ARGUMENTO
LOAD	* "n" unidad "nombre"

Ejemplos

- LOAD * "m"; 5; "Pelota"
- LOAD * "m"; 3; "ZX"

COMBINACION

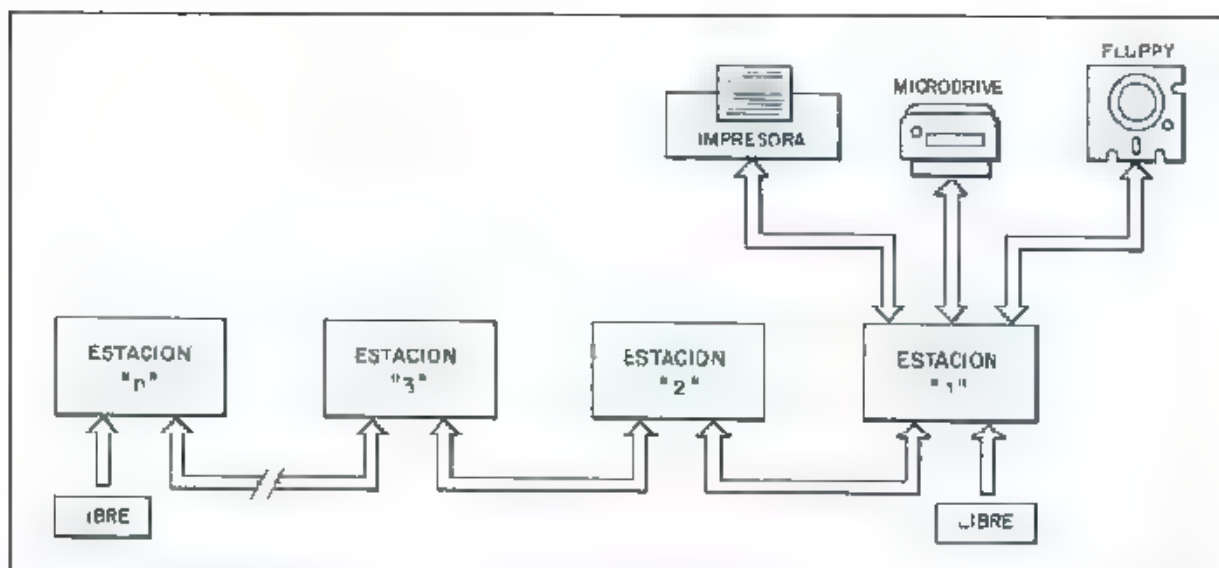
SENTENCIA	ARGUMENTO
MERGE	* "n" unidad "nombre"

Ejemplos.

- MERGE * "m", 7, "GO"
- MERGE * "m"; 2; "toro"

Borrado de programas

Una operación engorrosa es el borrado de programas en cassette, sin embargo, cualquier programa grabado en Microdrive puede ser borrado con facilidad utilizando el comando "ERASE", que se encarga, entre otras cosas,



Estructura de conexionado.

de modificar el catálogo para que no se visualice el nombre del programa borrado

ERASE

Acceso al teclado

WHITE



ERASE

Su estructura general es

SENTENCIA	ARGUMENTO
ERASE	"m" unidad "nombre"

Ejemplos.

- ERASE "m"; 1, "pepe"
- ERASE "m"; 5, "BANK"

Ejecución automática

Utilizando el Microdrive y siguiendo unas cuantas reglas podrá ejecutar de forma automática aquel programa

que más utilice de un cartucho.

El programa deberá grabarse de la siguiente forma

SAVE * "n", 1 "run" LNE n

donde "n" es el número de línea para su autoejecución.

Para utilizar la facilidad de ejecución automática tendrá que

- Insertar el cartucho en la unidad primera del Microdrive.

- Utilizar el programa después de haber conectado la alimentación del Spectrum o después de haber introducido el comando "NEW".

Teclee el comando "RUN" sin ningún argumento y el programa grabado con el nombre "run" se cargará en la memoria del Spectrum y se autoejecutará.

Protección de ficheros

Básicamente existen tres métodos para proteger sus programas.

a) Quitar la lengüeta de plástico situada en uno de los

laterales del cartucho. De esta manera no se podrán grabar o borrar ningún programa.

b) Grabar los programas a proteger con "LNE n". De esta manera no se podrá realizar "MERGE" para visualizar su contenido.

c) Al realizar la grabación de un fichero anteponga el código 0 al nombre

SAVE * "m", 1 CHR\$ 0 + "nombre"

Cuando intente obtener un catálogo del cartucho, el nombre de ese fichero no se visualizará, por tanto deberá recordarlo para poder cargarlo en otra ocasión

LOAD * "m", 1, CHR\$ 0 + "nombre"

Ficheros de datos

Aparte de los ficheros de programas, el usuario puede abrir y cerrar ficheros de datos tanto para grabación como para lectura.

Apertura de ficheros

Utilizando la sentencia "OPEN # " podemos asignar un fichero Microdrive a una corriente; si el fichero no existe, el sistema operativo interpreta que es de escritura; por el contrario, si existe, es de lectura

Por ejemplo:

```
OPEN # 5 "m" : "base"
```

Grabación de datos

Podremos grabar datos en un fichero siempre y cuando haya sido abierto para escritura

Para grabar datos en un fichero se utiliza la sentencia PRINT.

SENTENCIA	ARGUMENTO
PRINT	# corriente datos

Ejemplos.

- PRINT # 5; 20
- PRINT # 5; 3'7'8
- PRINT # 5; a\$
- PRINT # 5; 89'b\$'n\$

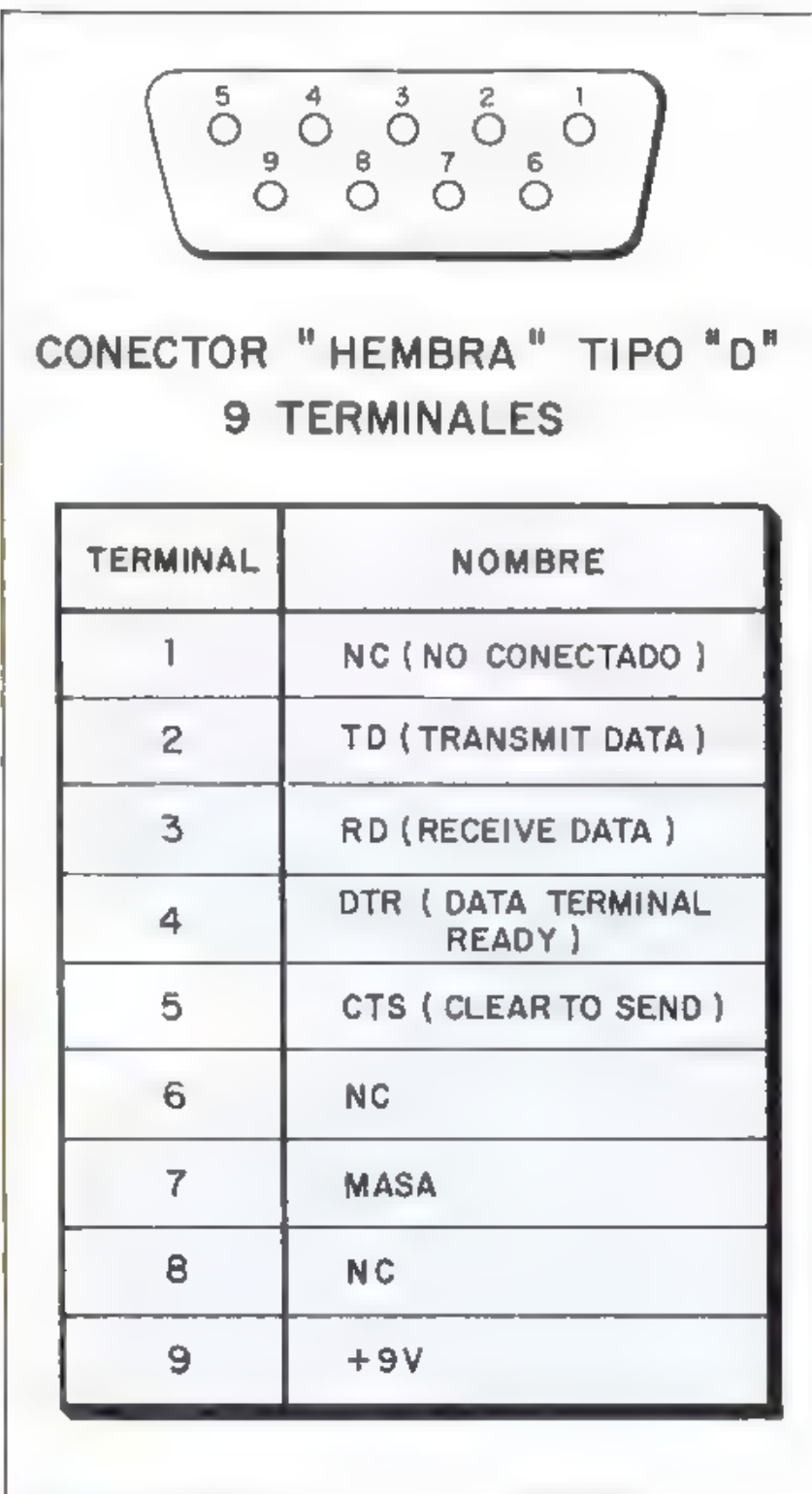
Para una posterior lectura, los datos deben ir seguidos del código "ENTER", para ello introduzcamos los datos uno a uno.

```
PRINT # 5 30
PRINT # 5 15
PRINT # 5
```

o utilice el signo del apóstrofe para separarlos

```
PRINT # 5 30'15'
```

Los datos no se graban directamente en el cartucho, si-



Conexión "RS 232".

no que se almacenan en una memoria intermedia de 512 bytes (0,5 Kbytes), cuando ésta se llena, se realiza la transferencia al Microdrive. La grabación se realiza por bloques.

Cierre de ficheros

Cuando la introducción de datos se termina, el fichero debe cerrarse, ya que de lo contrario no podría efectuarse una lectura posterior.

Al cerrar un fichero se transfieren al Microdrive los datos que estuvieran almacenados en la memoria intermedia

La forma de cerrar un fichero se realiza utilizando la sentencia "CLOSE #"

Ejemplo.

```
CLOSE # 5
```

Lectura de ficheros

Para leer datos de un fichero es necesario que este abierto para lectura

Los datos se leen utilizando la sentencia "INPUT"

SENTENCIA	ARGUMENTO
INPUT	# constante variables

```
INPUT #5; a
INPUT #5, b$
INPUT #5, c; n$; k
INPUT #5; LINE 1$
```

Si se leen varios datos seguidos, estos deben ir separados por "punto y coma"

```
INPUT #5 d, e, n, k$
```

En la lectura de datos de cadena es conveniente utilizar el siguiente formato.

```
INPUT #5, LINE C$
```

ya que si la cadena contiene comillas, interpretara que son el final de la cadena.

Ampliación de ficheros

Para realizar una ampliación de un fichero de datos ya creado, deberá seguir los siguientes pasos

- Abrir el fichero antiguo para lectura

CANALES DE COMUNICACION

"k" = Teclado

"s" = Pantalla

"p" = Impresora

"m" = Microdrive

"n" = Red de area local

"t" = RS232 Texto

"b" = RS232 Binario

CORRIENTES

#0 Salida y entrada de datos de la parte inferior de la pantalla.

#2 Salida de datos de la parte superior de la pantalla.

#3 Salida de datos a traves de la impresora.

#4-15 Libres para ser definidos por el usuario.

- Abrir un nuevo fichero para escritura

- Copiar el fichero antiguo en el nuevo.

- Editar a continuación los datos suplementarios en el fichero nuevo

- Cerrar los dos ficheros.

- Borrar el fichero antiguo

Ejemplo.

Tenemos un fichero llamado "DATOS" con 20 datos y deseamos incluir cinco más

```
10 84 32 15 y 49
```

a) Abrir fichero antiguo

```
OPEN #4 "m" 1 "DATOS"
```

b) Abrir fichero nuevo

```
OPEN #5 "m" 1 "DATOS 1"
```

c) Copiar fichero

```
FOR i = 1 TO 5
INPUT #4 dato
PRINT #5 dato
NEXT i
```

BASIC

```

ASTEROIDE
EDIT
ENSAMBLE
ESTELAR
G.D.U.
KILFREE
LISTADOR
MARTE.COD
PRONOFF
QUERTY
START
TRON.BAS
TRON.COD
VELOZ
ZZZ.ZZZ
    
```

24

Catálogo de un cartucho "microdrive".

d) Editar nuevos datos

PRINT # 5 10 84 37

PRINT # 5 5 44

e) Cerrar ficheros

CLOSE # 1

CLOSE # 2

f) Borrar fichero antiguo

ERASE "n" 1, "DATOS"

Programa

El programa numero "1" permite generar un fichero Bibliográfico de hasta 100 títulos de libros con sus correspondientes autores

Tanto los nombres de los libros como sus autores deberán tener una longitud máxima de 20 caracteres

Para salir de la opción "1" deberá teclearse la palabra

FICHERO

FECHA

LIBROS

21/06/85

1 LA REGENTA
CLARIN

2 1984
ORWELL

3 EL PADRINO
MARIO PUZO

4 RIMAS Y LEYENDAS
BECQUER

5 MACBETH
SHAKESPEARE

Programa "Fichero".

"FIN" como título de libro

La opción "2" permite grabar el fichero en un cartucho Microdrive. Los datos que debemos dar al programa son:

NOMBRE: nombre del fichero hasta seis caracteres, ya que el programa incluye cuatro más, como extensión del mismo

FECHA: Hasta un máximo de ocho caracteres

CLAVE: Tiene un máximo de diez caracteres, y permite proteger el fichero contra posibles lecturas de personas no autorizadas.

Los datos se graban en dos ficheros distintos

NOMBRE CLV
NOMBRE JAT

En el primero se graban los datos correspondientes al nombre del fichero, la fecha, la clave y el ultimo elemento almacenado

En el segundo los datos ge-

nerados.

La opción "3" permite leer un fichero generado anteriormente para su ampliación o visualización

Deberemos proporcionar el nombre sin extensión y la clave.

Si la clave almacenada en el fichero con extensión "CLV" no coincide con la teclada, el programa se autodestruye, no permitiendo el acceso a personas que no conozcan la clave

Por último la opción "4" permite visualizar un fichero, bien sea en pantalla o impresora

MOVE

Acceso al teclado

YELLOW

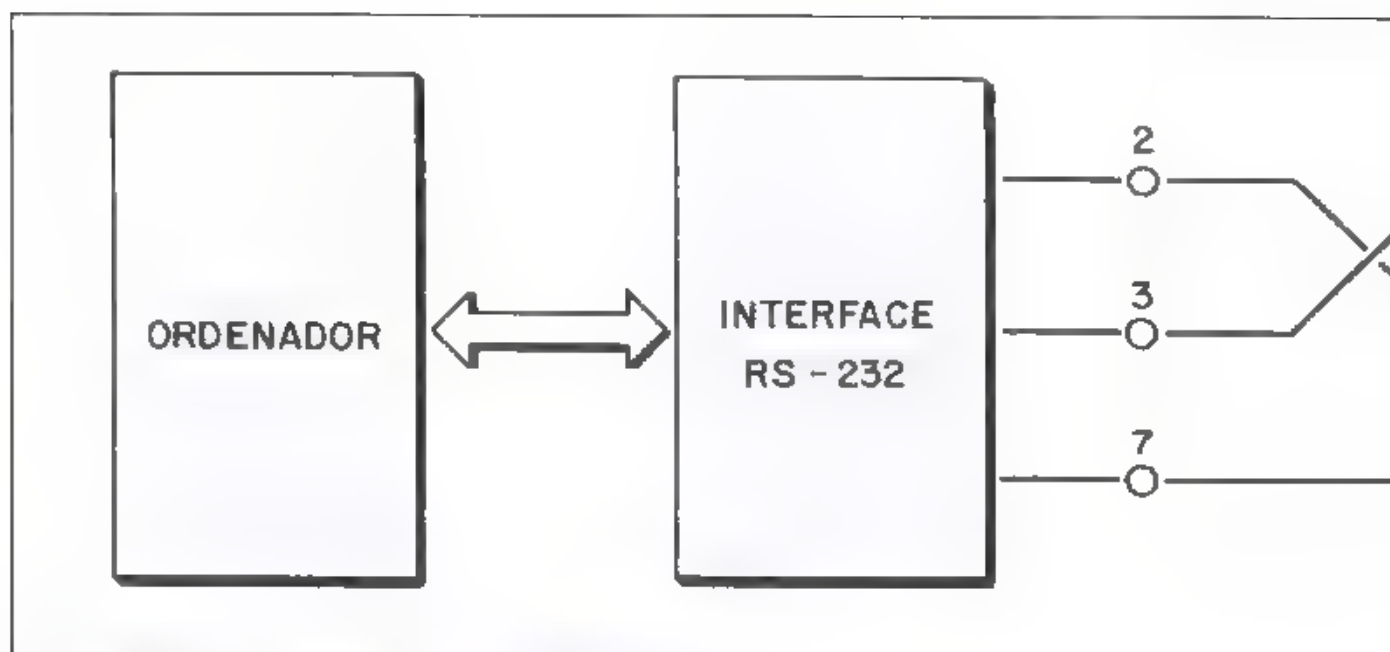


MODO E



MOVE

SYMBOL SHIFT



Conexión simple entre interfaces RS 232.

Definición

La sentencia "MOVE" transfiere los datos de un canal de comunicación (fuente) a otro distinto (destino).

La estructura general es:

SENTENCIA	ARGUMENTO
MOVE	canal TO destino

Ejemplos:

- MOVE # 1 TO # 2
- MOVE "m"; 1; "as" TO # 3
- MOVE "m"; 1; "Tn" TO "m"; 2, "Tn"
- MOVE # 1 TO # 3

La "fuente" y el "destino" pueden ser canales o números de corriente.

Esta sentencia es muy útil ya que se puede realizar

a) Copias de ficheros de datos

```

MOVE "n" 1 "CAN" TO "m"
1 "CANT"
MOVE "n" 1 "SET" TO "n" 2
"SET"
  
```

b) Visualización de ficheros

```
MOVE "m" 1 "TOR" TO # 2
```

c) Impresión de ficheros

```
MOVE "m" 1 "CAR" TO # 3
```

Con "MOVE" no se pueden realizar copias de programas, solamente de ficheros de datos.

"MOVE" se encarga de abrir y cerrar los ficheros, por tanto no es necesario incluir las sentencias "OPEN # " ni "CLOSE # ".

La red era local

La red de área local o LAN (Local Area Network) permite conectar hasta 64 Spectrum formando una red de comunicación.

Con el uso de la red se pueden establecer comunicaciones entre los diversos usuarios o estaciones.

Los periféricos comunes (Microdrive, impresora, floppy, etc.) pueden estar conectados a una sola estación, de

manera que todas las demás puedan hacer uso de ellos.

Una vez establecida la red, con sus correspondientes cables, es necesario identificar cada estación, para ello utilice la sentencia "FORMAT".

SENTENCIA	ARGUMENTO
FORMAT	"n" estación

Ejemplos:

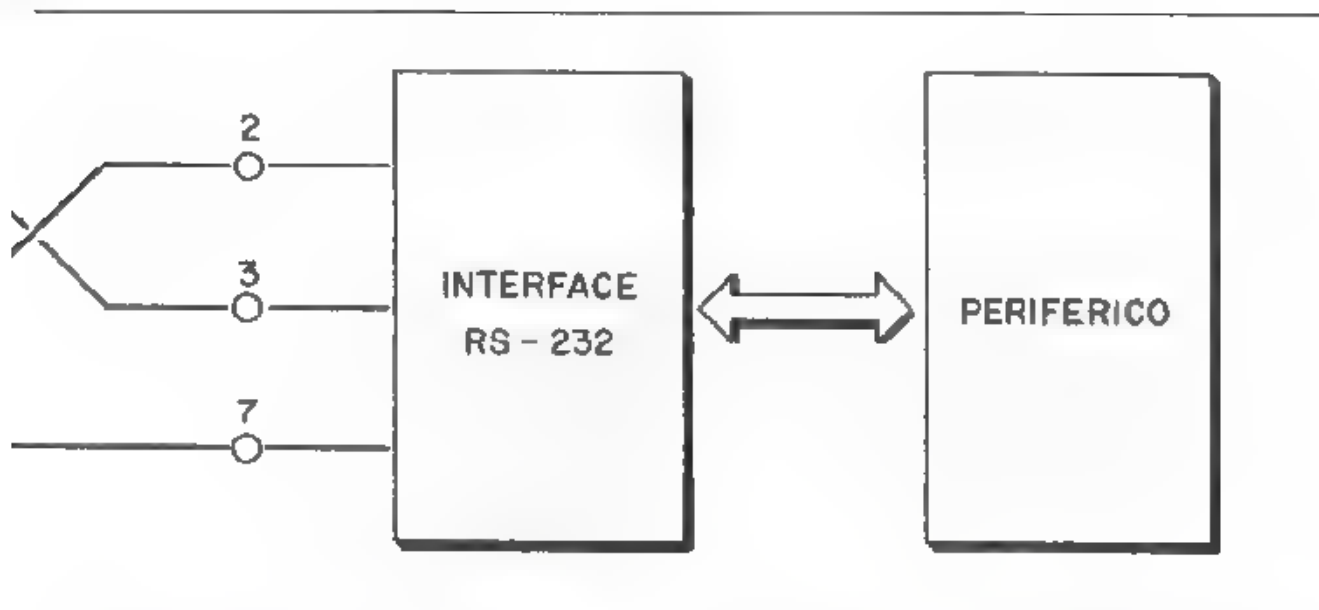
- FORMAT "n"; 2
- FORMAT "n"; VAL N\$
- FORMAT "n"; h
- FORMAT "n", a * 5

La cadena "n" es el indicativo del canal correspondiente a la red, y "estación" en un número entero entre 1 y 64.

Para enviar o recibir programas de la red se utilizan las sentencias "SAVE", "VERIFY", "LOAD" y "MERGE" con las siguientes variaciones.

```

SAVE * "n" estación destino
VERIFY * "n" estación fuente
LOAD * "n" estación fuente
MERGE * "n", estación fuente
  
```

Como podrá observar, la red no utiliza nombres de programas.

Cuando se desea establecer un diálogo entre una estación y todas las demás, simultáneamente, se utiliza el canal de difusión.

Las estaciones receptoras deben teclear

LOAD * "v" 0

y la transmisor

SAVE * "n" 0

Interface RS-232

Con el uso del interface RS-232, el Spectrum puede dialogar con cualquier periférico que utilice este protocolo de comunicación serie (Impresoras, modem, otro ordenador, etc.).

Es necesario realizar algunos ajustes en el periférico antes de utilizarlo.

— Desactivar el avance de línea automático.

— Desactivar el control de paridad

— Seleccionar ocho bits.

— Seleccionar un solo bit de parada

— Seleccionar la velocidad de transmisión, expresada en baudios (bits por segundo), entre cualquiera de las mostradas a continuación

50	2400
7.5	4800
300	9600
600	9200
1200	

El Interface RS-232 utiliza dos canales de comunicación

— El "t" para enviar textos en ASCII.

— El "b" para enviar códigos de ocho bits en binario.

Para utilizar cualquiera de los dos canales primero debe indicar al Spectrum la velocidad de transmisión a la que ha ajustado el periférico.

Ejemplos

```
FORMAT " " 2400
FORMAT "b", 600
```

El canal "t" envía el código ASCII de acuerdo al siguiente formato

Los códigos del 0 al 31 no se envían, excepto el código 13 (ENTER) que se envía seguido del 10 (cursor abajo) es decir, lo que se conoce por "retorno de carro" y "avance de línea".

— Los códigos del 32 al 127 ("espacio" al "copyright") se envían sin ninguna alteración

— Los códigos 128 al 164 (gráficos) no se envían, en su lugar se envía el código 63 (?)

— Los códigos del 165 al 255 (tokens) se expanden a los códigos 32-127.

Para enviar un listado a una impresora con Interface RS-232 y velocidad de transmisión de 300 baudios, se utilizaría

```
FORMAT " " 300
OPEN #5 "t"
LIST #5
```

Para recibir datos desde un terminal transmisor con una

velocidad de 9600 baudios, sería

```
10 FORMAT "" 9600
20 OPEN #5
30 PRINT NALYS # 5
40 GOTO 30
```

El canal "b" se utiliza para enviar los códigos de control a las impresoras, para utilizar un *modem*, etc

Con el canal «b» se pueden utilizar las sentencias "SAVE" y "LOAD".

```
SAVE * "r"
LOAD * "
```

dependiendo de si es transmisor o receptor.



El Spectrum, con el nuevo dispositivo (interface-1) y dos unidades de microdrive, puede acceder a 180 K de memoria externa.

Para enviar un código de control a una impresora, por ejemplo el 18 se utilizaría

```
OPEN # 7 : "L
PRINT # 7 CHR$ 18
CLOSE # 7
```

LA MEMORIA

Todo lo que introducimos en el ordenador, tiene que almacenarse dentro de alguna forma. Para ello todos los ordenadores disponen de lo que se denomina *memoria*.

Podemos imaginar la memoria de nuestro ordenador como un conjunto de 65536 cajitas, cada una de ellas puede contener un número comprendido entre 0 y 255. Cada cajita tiene, a su vez, un número que actúa como "nombre propio" y la distingue de las demás: la primera casilla se llama "0" y la última "65535". Este número se de-

nomina "Dirección" y el número que contienen se denomina "Dato".

El dato contenido en las primeras 16384 cajitas es fijo, no podemos alterarlo; es como si estas cajitas estuvieran cubiertas por un cristal, podemos ver el número que hay en su interior pero no podemos cambiarlo por otro.

En las restantes cajitas, si podemos alterar el contenido. Veamos ahora las sentencias de Basic que trabajan directamente sobre la memoria.

POKE

Acceso al teclado

PEEK



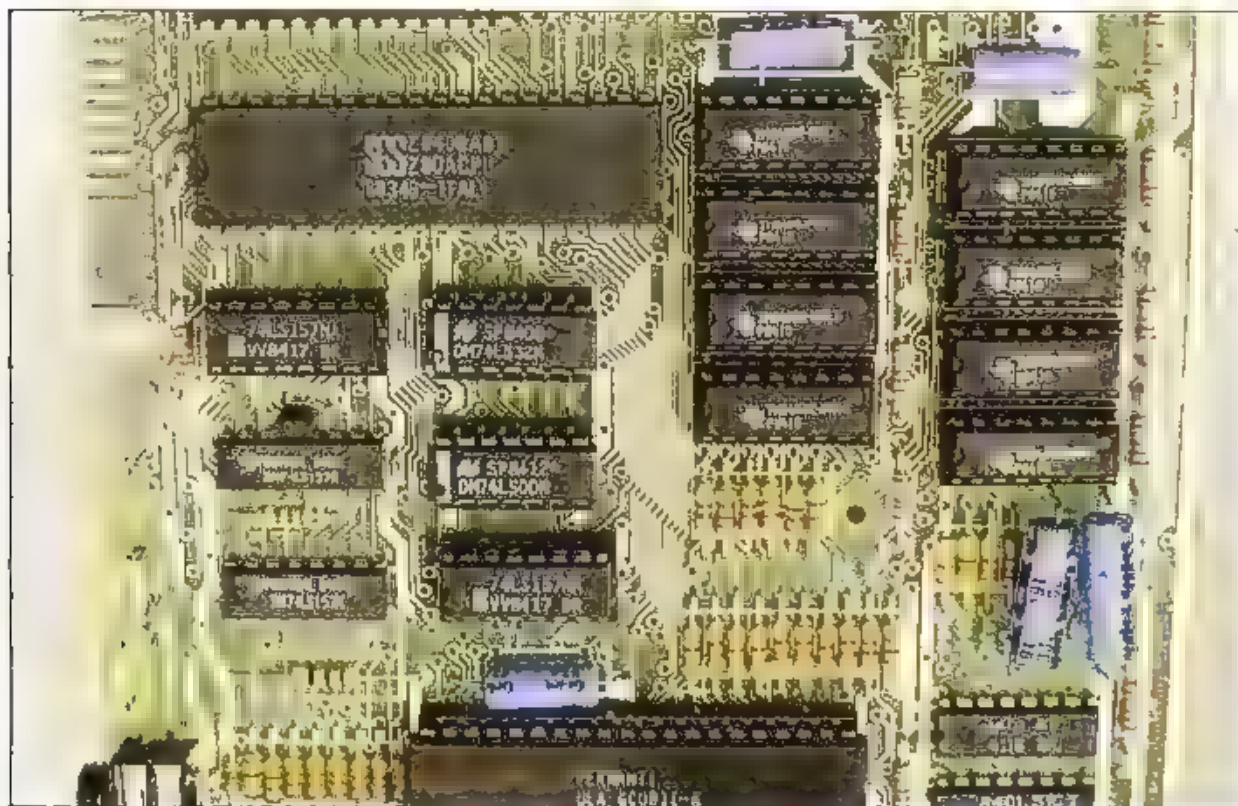
MODO K

OUT

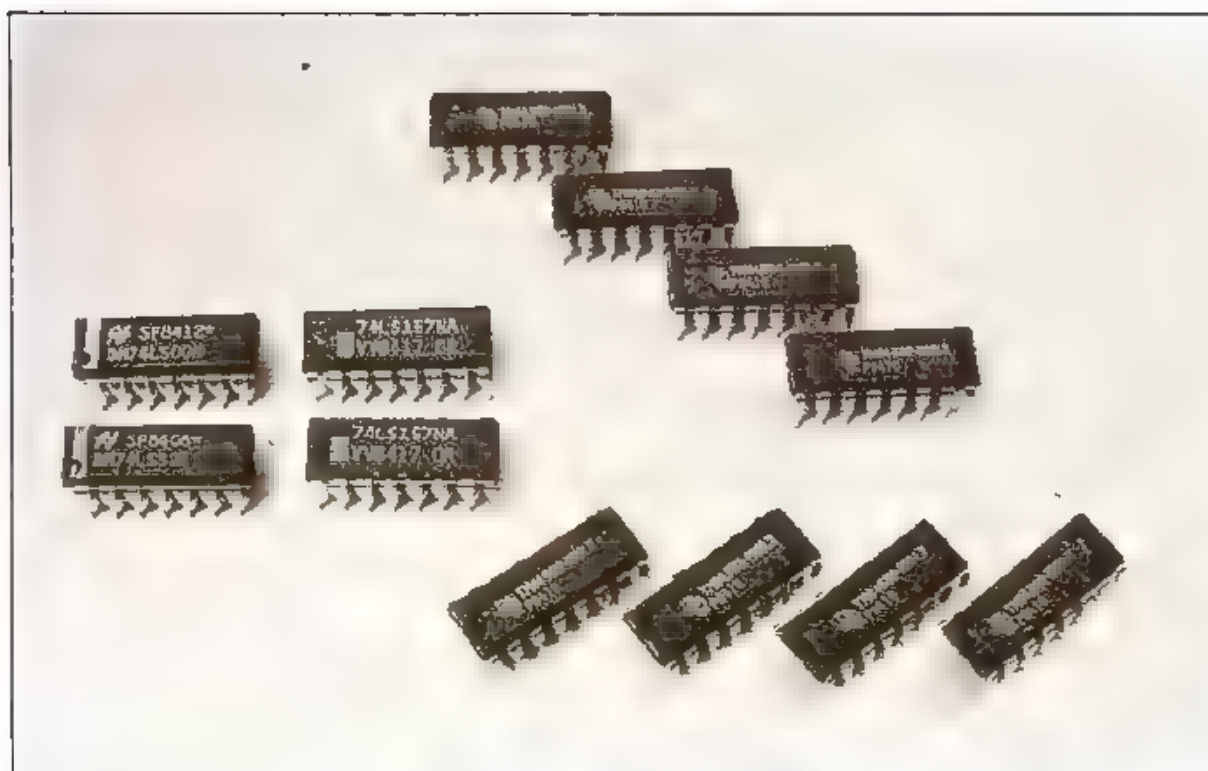
Definición

El comando POKE almacena un dato en una dirección de memoria.

Su estructura general es:



Los "chips" que componen la memoria del Spectrum.



Aspecto de la memoria del Spectrum compuesta por 16 chips de RAM dinámica y una de ROM.

SENTENCIA	ARGUMENTO
POKE	Dirección, Dato

Ejemplos:

- POKE 23300,255
- POKE 65342,a
- POKE d+38, 37
- POKE h-2, a-25

La dirección ha de estar comprendida entre 0 y 65535 (si bien, intentar "POKEar" en una dirección menor de 16384, no tiene sentido) y el dato ha de estar entre -255 y 255 (un dato negativo equivale a 256 menos ese número).

Si alguno de estos números estuviera fuera de este margen, se produciría el error:

B: illegal out of range

La razón de que no tenga sentido "POKEar" en direcciones inferiores a 16384 es que, como hemos dicho an-

tes, el contenido de estas direcciones no se puede alterar.

PEEK

Acceso al teclado

PEEK



MODOS E

OUT

Definición

La función PEEK devuelve el contenido de una posición de memoria, cuya dirección es el argumento de PEEK. Su estructura general es:

SENTENCIA	ARGUMENTO
PEEK	Dirección

Ejemplos

- PRINT PEEK 23657
- LET a=PEEK (60000+b)
- LET a=PEEK b
- PRINT 17+PEEK 56432

La dirección puede ser cualquier número entero comprendido entre "0" y "65535". Si estuviera fuera de este margen, se produciría el error.

B: illegal out of range

Las sentencias POKE y PEEK nos dan un inmenso poder sobre el ordenador al actuar directamente sobre la memoria, ya que todo lo que hace el ordenador depende de los datos almacenados en su memoria.

Tipos de memoria

Existen un gran número de

tipos de memoria, ROM, RAM, PROM, EPROM, EAROM, etc. Nuestro ordenador dispone de 16384 (16 K) direcciones o posiciones de memoria ROM y 49152 (48 K) de memoria RAM (en la versión de 16 K, son sólo 16384 posiciones de RAM).

ROM significa "Read Only Memory" (Memoria de sólo lectura), como su nombre indica, es una memoria que sólo se puede leer, y en la que no se puede escribir, su contenido ha sido grabado en fábrica de forma indeleble.

RAM significa «Random Access Memory» (Memoria de acceso aleatorio), en realidad, la ROM también es de acceso aleatorio, ya que podemos leer cualquiera de sus datos sin leer los precedentes, pero a la RAM se la llama así para distinguirla de la ROM, ya que en memoria RAM podemos tanto leer como escribir.

El mapa básico de memoria para ambas versiones sería el representado en la FIGURA 1.

Bit y byte

Hasta ahora hemos dicho que los números se almacenan en las posiciones de memoria, pero no hemos explicado la forma en que se almacena un número determinado.

Se puede imaginar que cada posición de memoria (cajita) es una fila de ocho interruptores, a un interruptor encendido le llamamos "1" y a uno apagado le llamamos "0". Cada número comprendido entre "0" y "255" se forma por combinación de "unos" y "ceros", es lo que se

posición de memoria almacena un byte

La correspondencia entre cada número y su combinación binaria, se haya sabiendo que cada bit tiene un valor determinado según su situación en el byte. Empezando por la derecha, el primer bit vale "1", el segundo vale "2", el tercero "4", y el cuarto "8", y así sucesivamente, cada uno vale el doble del anterior, hasta el octavo que vale "128". Para hallar el valor decimal de un número binario, se suman los valores de los bits que están a "1". Veamos unos cuantos ejemplos

128	64	32	16	8	4	2	1		
1	0	0	1	1	0	1	0	$128+16+8+2$	154
0	1	1	0	0	1	0	1	$= 64+32+4+1$	101
1	1	1	1	1	1	1	1	$= 128+64+32+16+8+4+2+1$	255
0	0	0	0	1		1	1	$= 8+4+2+1$	= 15

llama "Notación Binaria". A cada "1" o "0" se le denomina "Bit" (abreviatura de "Binary Digit"), y al conjunto de ocho "unos" o "ceros" se le denomina "Byte", por tanto, cada

Memoria ROM

El motivo de que la memoria ROM venga grabada de fábrica y sus datos no puedan ser

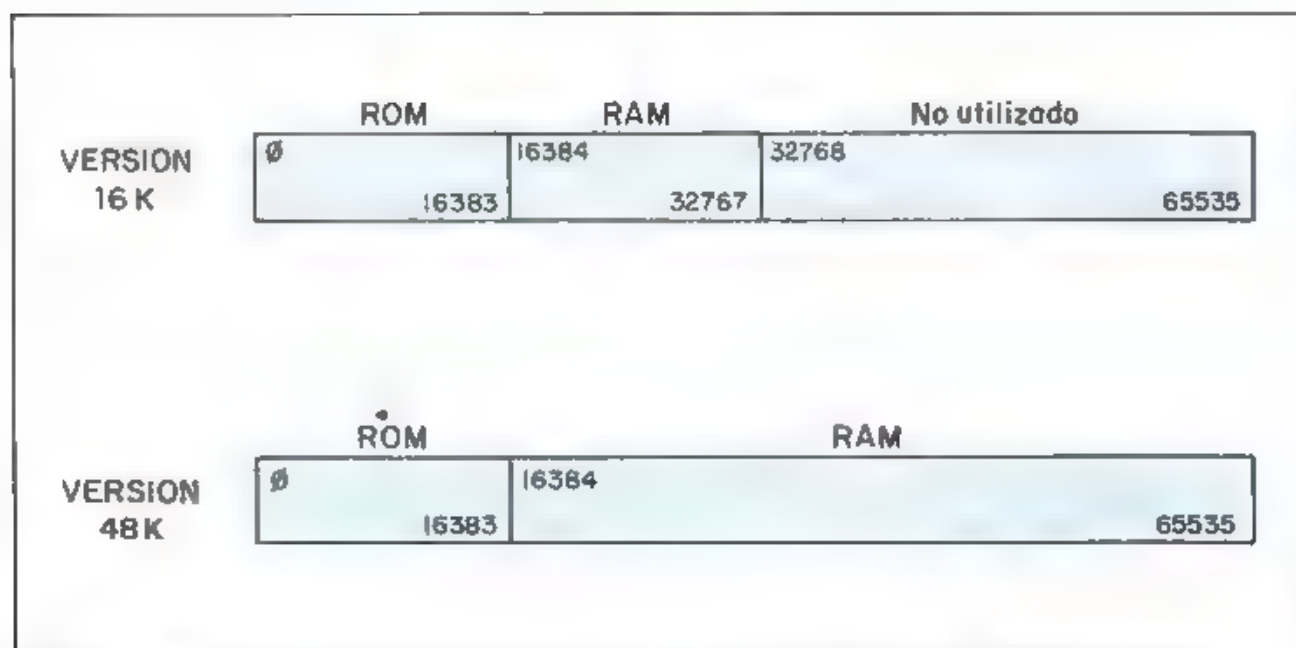


Figura 1. Mapa de memoria básico de ambas versiones del Spectrum.

		B I T S									
		7	6	5	4	3	2	1	0		
DIRECCION		128	64	32	16	8	4	2	1	CODIGO DECIMAL	
15880										0	
15881										60	
15882										66	
15883										66	
15884										126	
15885										66	
15886										66	
15887										0	

Figura 3. La letra "A" tal como está definida en la ROM del Spectrum

alterados, es que contiene lo que se denomina el "SISTEMA OPERATIVO" del ordenador, un conjunto de programas escritos en código máquina que permiten al ordenador operar desde el mismo momento en que se conecta.

También contiene la ROM el juego de caracteres, se encuentran a partir de la dirección 15616 y están definidos de la misma forma que los UDG. Como ejemplo, la letra "A" está definida de la forma que puede verse en la FIGURA 3.

Parte del Sistema Operativo del ordenador, lo constituye el denominado "INTERPRETE DE BASIC" que es precisamente, el programa que nos

permite utilizar este lenguaje para programar el ordenador.

La memoria RAM

En las direcciones que están a continuación de la ROM, se encuentra la memoria RAM. No toda ella está a nuestra disposición, ya que una parte la necesita el ordenador para sus propios datos.

Los primeros 6144 bytes (desde la dirección 16384 hasta la 22527) están ocupados por el archivo de presentación visual, en esta zona se encuentra almacenado todo lo que vemos en la pantalla.

Los 768 bytes siguientes

(desde la dirección 22528 hasta la 23295) contienen el archivo de atributos, que almacena los colores de todos los caracteres de la pantalla.

Entre la dirección 23296 y la 23551 (256 bytes) se encuentra la memoria intermedia de impresora. A continuación, entre la 23552 y la 23733 (182 bytes) están las variables del sistema.

A partir de la dirección 23734 se encuentra el área de información para canales; en la versión básica, esta zona ocupa 21 bytes, pero se expande al conectar el INTERFACE 1 y trabajar con MICRODIVE o con la ZX-NET.

A continuación, viene la zona donde almacenamos el

programa Basic y las variables, la longitud de estas zonas no es fija ya que se van expandiendo a medida que vamos almacenando datos en ellas. Las direcciones de inicio de cada una de estas zonas están contenidas en las variables del sistema CHANS, PROG y VARS que se actualizan continuamente por el sistema operativo.

Hay una serie de zonas más cuyas longitudes y direcciones de inicio tampoco son fijas, pero se encuentran anotadas en las correspondientes variables del sistema. Estas zonas son: Área de edición, zona de trabajo y pila del calculador.

A continuación viene una zona llamada de reserva, que contiene los bytes que no han sido utilizados, y por tanto, se contrae cada vez que se expande cualquiera de las otras zonas. Este área es considerablemente mayor en la versión de 48 K que en la de 16 K.

Todas las zonas que hemos visto hasta ahora se expanden hacia "arriba", pero hay dos zonas por encima del área de reserva que se expanden hacia "abajo", se trata de las pilas de máquina y GO SUB.

Finalmente, los últimos 168 bytes de la RAM se encuentran ocupados por los Gráficos Definidos por el Usuario (UDG).

En la FIGURA 2 se puede ver el mapa de memoria del Spectrum, los números situados a la izquierda son direcciones, y los situados a la derecha indican la cantidad de bytes ocupados por la zona correspondiente en el momento de conectar el ordenador.

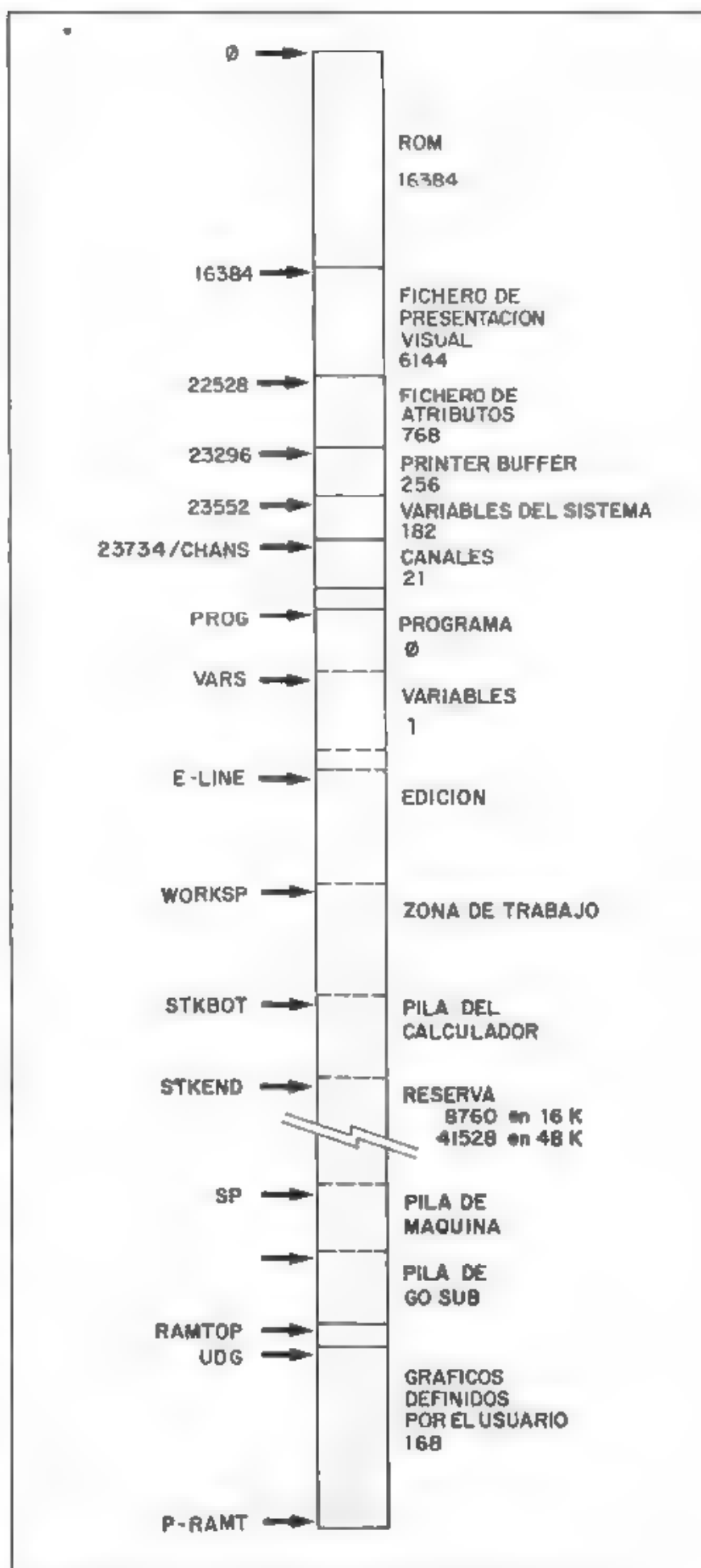


Figura 2. Mapa completo de memoria del Spectrum.

DIRECCION	DATO	
PROG	0	Número de linea
PROG + 1	10	
PROG + 2	12	Longitud de la linea + ENTER (12 bytes)
PROG + 3	0	
PROG + 4	241	Código de LET
PROG + 5	97	Código de "a"
PROG + 6	61	Código de "="
PROG + 7	50	Representación ASCII de "27"
PROG + 8	55	
PROG + 9	14	Indicador de número
PROG + 10	0	Representación en coma flotante de "27"
PROG + 11	0	
PROG + 12	27	
PROG + 13	0	
PROG + 14	0	Codigo de ENTER (fin de linea)
PROG + 15	13	

Figura 4. Una línea de programa en Basic, tal como se almacena en la zona de programa de la memoria RAM.

Almacenamiento de programas

Ahora vamos a ver como se almacenan en la memoria las líneas de Basic y las variables que introducimos cuando escribimos o cargamos un programa.

A partir de la dirección apuntada por la variable del sistema PROG, se almacena la primera línea del programa. En primer lugar, el número de línea que ocupa dos bytes (el más significativo primero); a continuación, viene la longitud de la línea que también ocupa dos bytes, pero con el orden invertido.

Los siguientes bytes constituyen el texto de la línea propiamente dicha, donde hay un número, está primero su representación ASCII, a continuación el código 14 y después la representación del número en coma flotante, que es la que realmente utiliza el ordenador.

Supongamos la siguiente línea:

```
10 LET a=27
```

En el interior de la memoria esta línea quedaría almacenada como se ve en la FIGURA 4.

Variables

La forma en que se almacenan las variables es algo más compleja, ya que depende del tipo de variable de que se trate.

El tipo de variable viene dado por la configuración de los tres primeros bits del primer byte, de la forma siguiente:

Variable numérica cuyo nombre es una sola letra: 011
Variable numérica cuyo nombre son varias letras: 101
Variable de cadena de caracteres: 010
Variable de control de bucle FOR-NEXT: 111
Matriz de números: 100
Matriz de caracteres: 110

A continuación, vamos a ver cada una detenidamente y con algún ejemplo.

Variable numérica cuyo nombre es una sola letra

Ejemplo

```
a=27
```

a	97	0 1 1 0 0 0 0 1
27	0	0 0 0 0 0 0 0 0
	0	0 0 0 0 0 0 0 0
27	0	0 0 0 1 1 0 1 1
	0	0 0 0 0 0 0 0 0
	0	0 0 0 0 0 0 0 0

Variable numérica cuyo nombre son varias letras

En este caso, la última letra tiene el primer bit a "1" para indicar que es el fin del nombre.

Ejemplo:

```
abcd=27
```

a	161	1 0 1 0 0 0 0 1
b	98	0 1 1 0 0 0 1 0
c	99	0 1 1 0 0 0 1 1
d	228	1 1 1 0 0 1 0 0
27	0	0 0 0 0 0 0 0 0
	0	0 0 0 0 0 0 0 0
27	0	0 0 0 1 1 0 1 1
	0	0 0 0 0 0 0 0 0
	0	0 0 0 0 0 0 0 0

Variable de cadena de caracteres

Los dos bytes que siguen al nombre indican la longitud. Ejemplo:

```
a$="HOLA"
```

A	65	0 1 0 0 0 0 0 1
Long	4	0 0 0 0 0 1 0 0
= 4	0	0 0 0 0 0 0 0 0
H	72	0 1 0 0 1 0 0 0
O	79	0 1 0 0 1 1 1 1
L	70	0 1 0 0 1 1 0 0
A	65	0 1 0 0 0 0 0 1

Variable de control de bucle FOR-NEXT

Los cinco primeros bytes que siguen al nombre indican el valor inicial en coma flotante, los cinco siguientes indican el límite, los cinco siguientes el "paso", los dos siguientes indican la línea donde se ha definido el bucle y el último indica el número de sentencia dentro de la línea.

Ejemplo:

```
0 FOR a=1 TO 7 STEP 2
```

a	238	1 1 1 0 1 1 1 0
valor	0	0 0 0 0 0 0 0 0
= 1	0	0 0 0 0 0 0 0 0
	1	0 0 0 0 0 0 0 1
	0	0 0 0 0 0 0 0 0
	0	0 0 0 0 0 0 0 0
límite	0	0 0 0 0 0 0 0 0
= 7	0	0 0 0 0 0 0 0 0
	7	0 0 0 0 0 1 1 1
	0	0 0 0 0 0 0 0 0
	0	0 0 0 0 0 0 0 0
paso	0	0 0 0 0 0 0 0 0
= 2	0	0 0 0 0 0 0 0 0
	2	0 0 0 0 0 0 1 0
	0	0 0 0 0 0 0 0 0
	0	0 0 0 0 0 0 0 0

línea = 10	10	0 0 0 0 1 0 1 0
sent = 1	1	0 0 0 0 0 0 0 1

Matriz de números

Los dos bytes siguientes al nombre, almacenan la longitud de la variable más sus dimensiones, el siguiente byte contiene el número de dimensiones, a partir de ahí y de dos en dos bytes, se almacenan las dimensiones y a continuación, los elementos en coma flotante, ordenados según los subíndices.

Ejemplo:

DIM A(2,2) LET a(1,1)=15
LET a(1,2)=26
LET a(2,1)=12 LET a(2,2)=7

a	129	1 0 0 0 0 0 0 1
ong. = 25	25	0 0 0 1 1 0 0 1
	0	0 0 0 0 0 0 0 0
dm. = 2	2	0 0 0 0 0 0 1 0
1 dim.	2	0 0 0 0 0 0 1 0
	0	0 0 0 0 0 0 0 0
2 dim.	2	0 0 0 0 0 0 1 0
	0	0 0 0 0 0 0 0 0
a(1,1) 15	0	0 0 0 0 0 0 0 0
	15	0 0 0 0 0 0 0 0
	0	0 0 0 0 1 1 1 1
	0	0 0 0 0 0 0 0 0
	0	0 0 0 0 0 0 0 0
a(1,2) = 26	0	0 0 0 0 0 0 0 0
	0	0 0 0 0 0 0 0 0
	26	0 0 0 1 1 0 1 0
	0	0 0 0 0 0 0 0 0
	0	0 0 0 0 0 0 0 0
a(2,1) = 12	0	0 0 0 0 0 0 0 0
	0	0 0 0 0 0 0 0 0
	12	0 0 0 0 1 1 0 0
	0	0 0 0 0 0 0 0 0
	0	0 0 0 0 0 0 0 0
a(2,2) = 7	0	0 0 0 0 0 0 0 0
	0	0 0 0 0 0 0 0 0
	7	0 0 0 0 0 1 1 1
	0	0 0 0 0 0 0 0 0
	0	0 0 0 0 0 0 0 0

Matriz de caracteres

Los dos primeros bytes que siguen al nombre, indican la longitud de la variable más sus dimensiones, el siguiente indica el número de dimensiones, a continuación vienen las dimensiones, y finalmente, el texto

Ejemplo:

DIM a\$(2,8) LET a\$(1)="BASIC"
LET a\$(2)="SINCLAIR"

a	193	1 1 0 0 0 0 0 1
ong. 21	21	0 0 0 1 0 1 0 1
	0	0 0 0 0 0 0 0 0
dm. = 2	2	0 0 0 0 0 0 1 0
1 dim.	2	0 0 0 0 0 0 1 0
	0	0 0 0 0 0 0 0 0
2 dim.	8	0 0 0 0 1 0 0 0
	0	0 0 0 0 0 0 0 0
B	66	0 1 0 0 0 0 1 0
A	65	0 1 0 0 0 0 0 1
S	83	0 1 0 1 0 0 1 1
I	73	0 1 0 0 1 0 0 1
C	67	0 1 0 0 0 0 1 1
" "	32	0 0 1 0 0 0 0 0
" "	32	0 0 1 0 0 0 0 0
" "	32	0 0 1 0 0 0 0 0
S	83	0 1 0 1 0 0 1 1
I	73	0 1 0 0 1 0 0 1
N	78	0 1 0 0 1 1 1 0
C	67	0 1 0 0 0 0 1 1
L	76	0 1 0 0 1 1 0 0
A	65	0 1 0 0 0 0 0 1
I	73	0 1 0 0 1 0 0 1
R	82	0 1 0 1 0 0 1 0

Borrado de variables

Ejecute el siguiente programa:

```
10 LET a=27
20 CLEAR
30 PRINT a
```

Verá que se detiene con el informe:

2 Variable not found, 30-1

La línea 30 no encuentra la variable "a" a pesar de haber sido definida en la línea 10. Lo que ocurre es que la variable ha sido borrada en la línea 20, ésta es una de las utilidades del comando CLEAR

CLEAR

Acceso al teclado

LEN



Definición

El comando CLEAR borra la pantalla, las variables, restaura la posición de PRINT a la esquina superior izquierda, restaura la posición de PLOT a la esquina inferior izquierda, restaura el puntero de DATA y, caso de tener un argumento numérico, cambia la dirección de RAMTOP si ello fuera posible.

Su estructura general es:

SENTENCIA	ARGUMENTO
CLEAR	Nueva RAMTOP

Vuelva a mirar la figura 2 donde se muestra el mapa de memoria, casi al final verá una dirección apuntada por una variable que se llama RAM-

TOP. Es la dirección más alta que puede utilizar el Basic, algo así como un "límite" del sistema.

Podemos variar la posición de este límite, dentro de unos márgenes, lo cual puede ser muy útil en determinados casos.

Normalmente, por encima de RAMTOP sólo se encuentran los Gráficos Definidos por el Usuario, pero podemos bajar la RAMTOP y hacer sitio para colocar algo que no deseemos que el Basic pueda borrar, por ejemplo, un programa en código máquina. Lo que coloquemos por encima de RAMTOP queda a salvo de borrados incluso con NEW.

Si desea saber en qué dirección se encuentra la RAMTOP de su ordenador, teclee:

```
PRINT PEEK 23730+256*PEEK 23731
```

La respuesta normal será 32599 para el modelo de 16 K, y 65367 para el de 48 K. Tenga en cuenta que algunos interfaces pueden alterar este valor.

Si desea bajar la RAMTOP teclee el comando CLEAR seguido de la nueva dirección. Por ejemplo, supongamos que su versión es de 16 K, en ese caso la RAMTOP estará en 32599. Supongamos ahora que quiere conseguir un espacio libre de 300 bytes, teclee:

```
CLEAR 32299
```

Ahora las direcciones libres van desde la 32300 hasta la 32599, ambas inclusive.

La dirección más baja que puede usar como argumento de CLEAR es 23821, y la más

alta es 32767 en la versión de 16 K y 65535 en la de 48 K. Si el argumento de CLEAR está fuera de este margen, obtendrá el informe:

```
M RAMTOP not good.
```

En este caso, CLEAR habrá hecho todo (borrar variables, restaurar punteros, etc.) menos cambiar la RAMTOP.

Cuando se llena la memoria

Es posible que haya tenido ya alguna experiencia de lo que ocurre cuando se llena la memoria de su ordenador, si jugamos un poco con CLEAR bajando mucho la RAMTOP, seremos capaces de verlo claramente. Si tiene el INTERFACE 1, desconéctelo para hacer estas pruebas. Teclee:

```
CLEAR 23850
```

Ahora intente pulsar cualquier tecla, verá que el ordenador no responde, y en su lugar, emite un «pitido» que dura un par de segundos. Hemos bajado tanto la RAMTOP que no le hemos dejado sitio al Basic para trabajar. No tendremos más remedio que desconectar el ordenador y volverlo a conectar de nuevo para que todo vuelva a la normalidad.

El «pitido» que hemos oído es la señal de alarma del Spectrum, indica que la memoria está totalmente llena. Ahora teclee:

```
CLEAR 23900
```

Ahora introduzca las siguientes líneas:

```
10 REM *****
20 REM *****
30 REM *****
```

No se asuste, si cuando ha ido a introducir la línea 30 el ordenador no la ha aceptado, y le ha respondido con el mensaje

```
G No room for line 01
```

Esto le indica que el área de Basic está llena, y no cabe ninguna línea más. Ahora haga NEW y teclee:

```
10 REM *****
20 REM *****
```

Esta vez, no sólo no ha podido introducir la línea 20, sino que ni siquiera ha podido terminarla, cuando llevaba 22 "x" pulsadas e iba a pulsar la 23, el ordenador se ha bloqueado de nuevo con el «pitido», la memoria está tan llena que no hay sitio suficiente en el área de edición para construir la nueva línea. Borre toda la línea 20 pulsando DELETE, e introduzca:

```
CLEAR 24000 NEW
```

Ahora teclee:

```
DIM a(10,0)
```

Al pulsar ENTER, el ordenador le responderá con el

RENUMERADOR - LISTADO ASSEMBLER

Direcc.	Cód. Mq.	Listado Assembler
23296	42,83,92	LD HL, (PROG)
23299	17,10,0	LD DE, +0A
23302	229	BUCLE PUSH HL
23303	237,75,75,92	LD BC, (VARS)
23307	237,60	SBC HL, BC
23309	40,10	JR Z, (FINAL)
23311	225	POP HL
23312	114	LD (HL), D
23313	35	INC HL
23314	115	LD (HL), E
23315	75	INC HL
23316	79	LD C, (HL)
23317	75	INC HL
23318	70	LD B, (HL)
23319	9	ADD HL, BC
23320	35	INC HL
23321	235	EXX DE, HL
23322	1,10,0	LD BC, +0A
23325	9	ADD HL, BC
23326	235	EXX DE, HL
23327	24,220	JR (BUCLE)
23329	207	FINAL RST 00
23330	255	DEFB +FF
		PROG EQU +5C53
		VARS EQU +5C40

Figura 5. Un sencillo renumerador en Código Máquina para el Spectrum.

mensaje.

4 Out of memory. 0 1

Hemos intentado dimensionar una matriz de 10 por 10, lo que requiere 508 bytes disponibles en el área de variables, pero tenemos tan baja la RAMTOP que no hay espacio suficiente.

Este mensaje se presentará cada vez que intente hacer algo para lo que necesite más memoria que la que tiene disponible.

Siempre que utilice CLEAR en un programa, recuerde que además de bajar la RAMTOP, le borrará la pantalla y todas las variables que hubie-

ra definido hasta ese momento, además de restaurarle las posiciones de PRINT y PLOT y el puntero de DATA, de la misma forma que si hubiera ejecutado un RESTORE y un CLS.

Puede utilizar CLEAR sin argumento, que hará todo excepto modificar la RAMTOP

Programando en código máquina

El Spectrum, al igual que la mayoría de los ordenadores, permite llamar desde el Basic a rutinas escritas en código máquina. El código máquina no es realmente un lenguaje de programación (el lenguaje correspondiente es el Assembler) sino el conjunto de números que, almacenados en las posiciones de memoria, le indican al microprocesador las operaciones que debe ir ejecutando.

En lenguajes de alto nivel, como el Basic, cada comando desencadena la ejecución de cientos de instrucciones en código máquina, pero puede haber cosas que no se pueden hacer en Basic, o que se hacen más deprisa en có-

digo máquina, para ello se ha previsto la función **USR**

JSR

Acceso al teclado

USR



MODOS E

ATTR

Definición

La función **USR** con argumento numérico, ejecuta las instrucciones en código máquina correspondientes a los

números almacenados a partir de la dirección apuntada por el argumento, hasta que se encuentre una instrucción **RET** (código 201), momento en el que devuelve como resultado el contenido del par de registros **BC** del microprocesador.

Su estructura general es:

FUNCION	ARGUMENTO
USR	Dirección

Los programas en código máquina, normalmente, se escriben primero en Assembler, y luego se traducen por medio de un programa que se conoce con el nombre de "Ensamblador".

En este caso, el mismo ensamblador se encarga de introducir el programa en el ordenador. No obstante, si no



La perfección alcanzada en los juegos comerciales sólo es posible con un dominio absoluto del código máquina.

se dispone de ensamblador, también es posible hacer la traducción "a mano". En este caso, será necesario escribir un pequeño programa en Basic que se encargue de introducir el código máquina que, normalmente, se encontrará en sentencias DATA.

Hay dos formas de almacenar un programa en código máquina, una es bajar la RAMTOP (con CLEAR) y almacenar el programa por encima de ésta, con lo que quedará a salvo de borrados accidentales. Este es el sistema más usado, pero en determinados casos, puede ser interesante meter un programa corto en la memoria intermedia de impresora, si bien hay que tener en cuenta que, en este caso, será borrado por cualquier comando que utilice la impresora, o bien por el comando NEW

Para ilustrar la velocidad y posibilidades del código máquina, hemos desarrollado una pequeña rutina que permite renumerar las líneas del programa Basic, empezando por la línea 10 y siguiendo de 10 en 10.

En la FIGURA 5 se muestra el listado del programa en lenguaje Assembler, a la izquierda está la traducción a código máquina.

En este caso hemos preferido almacenar el programa en la memoria intermedia de impresora, con el fin de que las direcciones sean las mismas para 16 y 48 K. No obstante, el programa es "reubicable", lo que quiere decir que puede correr en cualquier posición de memoria.

El siguiente programa en Basic se encarga de introducir el código máquina en memoria

```
10 LET dir=23296
20 FOR n=1 TO 35
30 READ a:POKE dir,a
40 LET dir=dir+1
50 NEXT n
60 DATA 42,83,92,17,10,0,229,2
37,75,75,92,237,86,40,10,225,114
35,115,55,76,35,76,9,35,235,1,1
4,0,9,233,24,229,227,255
```

Cuando lo haya ejecutado, puede salvarlo en cinta teclando:

```
SAVE "ramm" "CODE 23296,35"
```

Cuando tenga un programa Basic en el que las líneas no estén numeradas de 10 en 10, cargue este renumerador con:

```
LOAD "CODE"
```

Y cuando lo tenga, teclee:

```
RANDOMIZE USP 23296
```

Deberá obtener el mensaje:

```
0 OK, 0 1
```

que le indicará que todo ha ido correctamente; si ahora hace un listado, verá que las líneas están numeradas de 10 en 10 y empezando por la 10. Tenga en cuenta, no obstante, que los GO TO y GO SUB no habrán sido renumerados, por lo que deberá hacerlo manualmente.

Si desea que la primera línea sea la 100 y que se numeren de 50 en 50, teclee.

```
POKE 23300,100
POKE 23323,50
```

En general, la dirección 23300 almacena el número de la primera línea (entre 0 y 255) y la dirección 23323 el incremento (también entre 0 y 255).

Si desea que un programa en código máquina se auto-ejecute, deberá utilizar un pequeño cargador en Basic de la forma.

```
10 LOAD "CODE RANDOMIZE
USP (direccion)
```

que salvará en cinta con SAVE... LINE 10 antes del programa en código máquina. ■

LOS PERIFERICOS

Un ordenador se compone, básicamente, de una CPU (Unidad Central de Proceso) y de una cierta cantidad de memoria. En el Spectrum, la CPU es el microprocesador Z-80. Este núcleo debe comunicarse con el exterior, para lo cual se sirve de los dispositivos periféricos.

Un periférico es todo dispositivo que se une al ordenador, excepto la CPU y la memoria. Son ejemplos de periféricos, el teclado, la pantalla, el cassette, el joystick, la impresora, el Microdrive, etc.

La CPU se comunica con la memoria a través de los buses de direcciones y de datos, indicando en el bus de control, que quiere acceder a la memoria. Igualmente, para comunicarse con los periféricos utiliza los buses de direcciones y datos, pero esta vez, el bus de control indica que se está accediendo a un periférico. La pantalla es una excepción, ya que la comunicación se hace mediante la ULA que funciona como una segunda "pseudo-CPU" con mayor prioridad.

De la misma forma que cada posición de memoria tiene una dirección, los periféricos tienen también uno o varios números que los definen. Estos números se denominan "ports" (en inglés, "puertos") y cumplen la misma función que las direcciones en la me-



Entre los sistemas de almacenamiento de datos alternativos al cassette se encuentra el wafadrive.



El joystick es especialmente adecuado para los juegos.

moria. Aunque el Z 80 sólo puede direccionar 256 ports de entrada/salida, el Spectrum se las arregla de forma ingeniosa para trabajar con números de port superiores a 255

Las instrucciones que envían y reciben datos a y desde los ports tienen una sintaxis muy similar a las de la memoria (POKE y PEEK). Vamos a verlas a continuación.

OUT

Acceso al teclado

PEEK

MODO E



+

SIMBOL
SHIFT

OUT

Definición

El comando OUT escribe un dato en un port de salida, el número de port se indica mediante la dirección.

Su estructura general es:

SENTENCIA	ARGUMENTO
OUT	Dirección, Dato

Ejemplos:

- OUT 254,16
- OUT d,27
- OUT 25+d,a
- OUT 254,a+8

La dirección ha de estar comprendida entre 0 y 65535, y el dato, ha de estar entre -255 y 255 (un número negativo equivale a 256 menos ese número).

Si alguno de estos números estuviera fuera de este margen, se produciría el error:

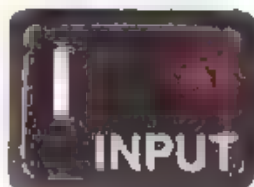
B Integer out of range

IN

Acceso al teclado

CODE

MODO E



+

SIMBOL
SHIFT

IN

Definición

La función IN tiene como argumento la dirección de un port, y devuelve como resultado el dato que se encuentra en ese momento en el port.

Su estructura general es:

SENTENCIA	ARGUMENTO
IN	Dirección

Ejemplos:

- PRINT IN 254
- LET a=IN 32766
- PRINT 27+IN 223
- LET a=IN b

La dirección puede ser cualquier número entero comprendido entre 0 y 65535. Si estuviera fuera de este margen, se produciría el error:

B Integer out of range

En el Spectrum no se puede utilizar arbitrariamente cualquier port, cada uno tiene su función concreta y existen

números de port que no tienen ningún sentido. Para comprender el funcionamiento de los ports es imprescindible atender a la configuración binaria de los buses de direcciones y datos cuando se llama a cada port.

El bus de direcciones está compuesto por 16 bits, y cada uno maneja un determinado periférico. Los bits se numeran del 0 al 15 empezando por la derecha, y precedidos de una "A" para indicar que se trata del bus de direcciones. Así el bit de más a la derecha se denomina "A0", el siguiente "A1", y así sucesivamente hasta el de más a la izquierda que se denomina "A15".

La configuración binaria que se produce en el bus de direcciones, traducida a decimal, constituye la dirección del port, por ejemplo, la configuración binaria 00000000 01111110 corresponde al port 254 (uno de los más usados en el Spectrum). El bus de datos está compuesto por 8 bits, la configuración binaria de estos bits, traducida a decimal, constituye el dato que se almacena en el port o que se lee del mismo.

Los ocho bits de la derecha del bus de direcciones (A0 a A7), indican a qué periférico se quiere acceder. Esto se indica poniendo este bit a "0" mientras que los demás permanecen a "1", sólo uno de estos bits debe ser "0" a la vez, ya que de lo contrario, se podría crear confusión en la ULA al intentar acceder a varios periféricos simultáneamente.

Los ocho bits de la izquierda (A8 a A15) deben ser normalmente cero, sólo se utilizan cuando se desea acceder al teclado, en este caso,

IN 65278	L LIST V / CLS FLASH	LPRINT C ? CONT PAPER	EXP X £ CLEAR INK	LN Z : COPY REEP	 CAPS SHIFT
IN 65022	ABS G THEN BOTO	BGN F TO FOR	DATA D STEP DIN	RESTORE S NOT SAVE	READ A STOP NEW
IN 64510	RND T > RAND MERGE	INT R < RVN VERIFY	TAN E >= REM ATN	COS W <> DRAW ACS	SIN Q <= PLOT ASN
IN 63486	CYAN 5 % CLOSE #	GREEN INV. VIDEO 4 \$ OPEN #	MAGENTA TRUE VIDEO 3 LING	RED CAPS LOCK 2 FN	BLUE ED T 1 DEF FN BLACK DELETE
IN 61438	YELLOW 6 & NOVE	WHITE 7 / ERASE	 8 (POINT	GRAPHICS 9) CAT	 0 _ FORMAT
IN 57342	STR \$ Y AND RETURN (CHR \$ U OR IF)	CODE I , AT INPUT IN	PEEK O ; POKE OUT	TAB P " PRINT ©
IN 49150	SOR H ↑ GDSUB CIRCLE	VAL J - LOAD VAL \$	LEN K + LIST SCREEN \$	USR L = LET ATTR	 ENTER
IN 32766	SIN B * BRIGHT	INKEY \$ N _ OVER	PI M INVERSE	 SYMBOL SHIFT	 BREAK SPACE
	239 (175)	247 (183)	251 (187)	253 (189)	254 (190)

Figura 1. Los ports del teclado en el Spectrum.

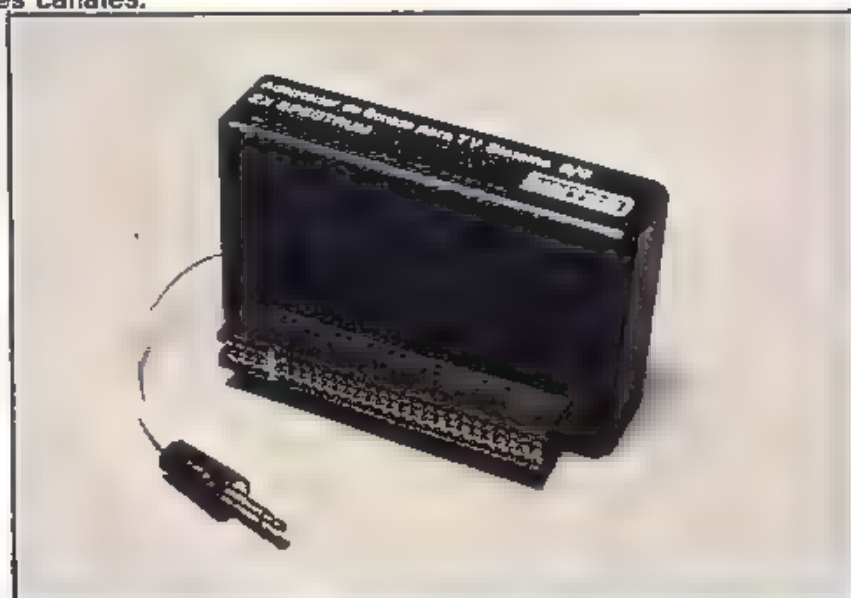


Las posibilidades sonoras del Spectrum quedan notablemente mejoradas con el uso de este sintetizador musical de tres canales.

cada uno indica una semifila (las cinco teclas derechas o izquierdas de una fila horizontal). El bit correspondiente a la fila deberá ser "0" mientras que los demás permanecerán a "1".

En la FIGURA 1 podemos ver las ocho semi-filas del teclado, a la izquierda está el número de port que se utiliza para leer cada semifila. Este dato varía según se trate de un modelo "ISSUE 2" o "ISSUE 3B". Los datos representados entre paréntesis corresponden al "ISSUE 3B" (Spectrum Plus). Si no hay ninguna tecla pulsada, el dato obtenido sería 255 en el "ISSUE 2" y 191 en el "ISSUE 3B".

En la TABLA 1 se ve la configuración binaria del bus de direcciones para cada uno de estos ports. Finalmente, el



La señal de ancho del Spectrum puede mezclarse con la de video para ser reproducida simultáneamente en T.V.

PROGRAMA 1 es un bucle que permite leer las ocho semi-filas devolviendo los datos de cada una en el "array" a (8) y presentándolos ordenados en la pantalla

65278	11111110	11111110
65022	11111101	11111110
64610	11111011	11111110
63488	11110111	11111110
61438	11101111	11111110
57342	11011111	11111110
49150	10111111	11111110
32766	01111111	11111110

```

10 DIM a(8) PRINT AT 8,0.
20 FOR n=0 TO 7
30 LET a(n+1)=IN (254+255*(255
270
40 PRINT a; n+1, "I=" a(n+1)
50 NEXT n GO TO 10

```

Ejecute el PROGRAMA 1 y pulse varias teclas, le servirá para ver qué dato entrega cada una en su port correspondiente

El port 254 es, sin duda, el más usado en la versión básica del Spectrum, la función IN 254 nos sirve para leer la entrada EAR, la señal está presente en el bit D6 del bus de datos

Si lo utilizamos como salida, podremos controlar el color del borde con los bits D0, D1 y D2 del bus de datos, ejecute el siguiente programa:

```

10 INPUT Color del borde " a
20 OUT 254,a GO TO 10

```

Tenga en cuenta que el color del borde es temporal, por lo que desaparecerá al pulsar cualquier tecla. Con el siguiente programa podrá conseguir un efecto curioso:

```

10 FOR n=0 TO 7 OUT 254,n
20 NEXT n GO TO 10

```

El port 254 sirve también para hacer sonar el altavoz interno, la señal deberá estar presente en el bit D4. Por últi-

mo, el bit D3 excita la salida MIC.

Otro port muy importante del Spectrum es el 223 ya que nos sirve para leer un joystick tipo Kempston. Si tiene un interface joystick de este tipo, ejecute el PROGRAMA 2. Se trata de un "Tele-Sketch" con el que podrá dibujar por la pantalla, también puede borrar si mantiene apretado el pulsador de "disparo".

En la versión básica del Spectrum, los bits A5, A6 y A7 del bus de direcciones no se usan, por lo que podrá utilizarlos si decide construir su propio interface. Las direcciones adecuadas serán aquellas que hagan que A0-A4 sean todo "unos", estas direcciones son: 31, 63, 95, 127, 159, 191, 223 y 255.

PROGRAMA 2

```

10 REM PROGRAMA 2
20 LET x=128 LET y=88 LET i=
1
100 PLOT INVERSE i,x,y
110 LET a=IN 223 GO TO 150-32*
(a>0)
120 LET i=(a>16) LET a=a-16*(
a>16)

```

```

130 LET x=x+1((a=1 OR a=9 OR a=5
) AND x<255)-(a=2 OR a=10 OR a=
6) AND x>0)
140 LET y=y+1((a=8 OR a=9 OR a=1
0) AND y<175)-(a=4 OR a=5 OR a=
6) AND y>0)
150 PLOT x,y
160 GO TO 100

```



Joystick Kempston y un poco de paciencia.

VARIABLES DEL SISTEMA

De la misma manera que un programa Basic utiliza una serie de variables, el Sistema Operativo (que de hecho es un programa escrito en código máquina) utiliza las suyas; son lo que se denomina "Variables del Sistema".

Las Variables del Sistema están todas juntas, y ocupan direcciones de memoria fijas. Tienen nombres, pero el ordenador no los reconoce, su única finalidad es servir a efectos nemotécnicos, para recordarnos su función. El verdadero nombre por el que se hace referencia a una variable en concreto es la dirección de la posición de memoria que ocupa. La TABLA 1 es

una lista de todas las variables del sistema ordenadas alfabéticamente, con su dirección en decimal y hexadecimal y el número de bytes que ocupan.

Cuando una variable ocupa más de un byte, el primero contiene el octeto menos significativo y el último, el más significativo, por ejemplo, si el contenido de una variable de dos bytes de longitud fuera "3B4C" (en hexadecimal), el primer byte contendría "4C" y el segundo "3B". Justo al revés de lo que parecería normal, pero éste es el formato que necesita el microprocesador para poder leer los números correctamente.

La mayor parte de las variables ocupan dos bytes. Si desea leer el contenido de una variable cuya dirección es "d", utilice:

```
PRINT PEEK d+256*PEEK (d+1)
```

Y si desea almacenar el número "n" en una variable cuya dirección es "d", utilice:

```
POKE d,n-256*INT (n/256)
POKE d+1, INT (n/256)
```

El PROGRAMA 1 sirve para imprimir el contenido de cualquier variable del sistema, para ello pregunta primero el nombre de la variable, que deberá teclearse tal como aparece en la TABLA 1.

PROGRAMA 1

```
10 REM *****
*          LECTURA DE          *
*          VARIABLES DEL        *
*          SISTEMA              *
*          *****
20 DEF FN a(d)=PEEK d+256*PEEK
(d+1)
30 DEF FN b(d)=FN a(d)+65536*P
EEK (d+2)
100 REM CARGA DATOS
110 DIM a$(65,5) DIM b$(65,5)
DIM c$(1,6) RESTORE 8000 CLS
120 FOR n=1 TO 65 READ a$(n)
NEXT n
130 FOR n=1 TO 65 READ b$(n)
NEXT n
200 REM PIDE VARIABLE
210 POKE 23658,8
220 INPUT "Que variable desea e
xplorar? ",c$(1)
230 PRINT AT 19,0,"
240 POKE 23658,0
250 IF c$(1)="NINGUN" THEN CLS
GO TO 9999
300 REM BUSCA VARIABLE
310 FOR n=1 TO 65 IF a$(n)=c$(
1) THEN GO TO 340
```

```
320 NEXT n
330 LET t=0 GO TO 400
340 LET t=VAL b$(n,1) LET d=VA
L b$(n,2) TO 1
400 REM DISTRIBUYE A RUTINAS
410 IF NOT t THEN GO TO 490
420 GO TO t+1000
450 PRINT AT 19,0, FLASH 1.
ESA VARIABLE NO EXISTE
GO TO 200
500 REM IMPRIME TIPOS 1, 2 Y 3
510 GO SUB 7000 PRINT "Se usa
momento vale ",v: GO TO 200
1000 REM VARIABLE TIPO 1
1010 LET v=PEEK d GO TO 500
2000 REM VARIABLE TIPO 2
2010 LET v=FN a(d) GO TO 500
3000 REM VARIABLE TIPO 3
3010 LET v=FN b(d) GO TO 500
4000 REM VARIABLE TIPO 4
4010 LET v=PEEK d LET v1=PEEK (
d+1)
4500 REM IMPRIME TIPO 4
5010 GO SUB 7000 PRINT "tiene d
os valores: ",v," y ",v1
510 GO TO 200
5000 REM VARIABLE TIPO 5
5010 GO SUB 7000 PRINT "Se usa
para leer el teclado. y tiene d
inco valores que son
5520 FOR i=0 TO 4 PRINT KSTATE
",i, ",PEEK (d+i) NEXT i
```



```

GO TO 200
6000 REM VARIABLE TIPO 6
6500 REM IMPRIME TIPO 6
6510 IF 3 < 7000 PRINT "LÍNEAS DE
6 DE LAS CORRIENTES
6520 FOR n=0 TO 37 STEP 2 PRINT
FN a d+n) NEXT n GO TO 200
7000 REM CABECERA DE IMPRESION
7010 PRINT "VARIABLES"
1) DIRECCION d RETURN
8000 REM LISTA DE VARIABLES
8010 DATA "ATTR P" "ATTR T" "BOR
OCR" "BREG" "CH ADD" "CHANS" "CH
ARS" "COORDS" "CURCHL" "DATADD
8020 DATA "DEFADD" "DEST" "DF CC
" "DF SZ" "DFCCL" "E LINE" "E PP
C" "ECHO E" "ERR NR" "ERR SP"
8030 DATA "FLAGS" "FLAGS2" "FLAG
X" "FRAMES" "K CUR" "K DATA" "K
TATE" "LAST K" "LISTSP" "MASK P"
8040 DATA "MASK T" "MEM" "MEMBOT
" "MODE" "NELPPC" "NMI" "NSPPC"
"NEXTLIN" "OLDPPC" "OSPPC"
8050 DATA "P FLAG" "P POSN" "P R
AMT" "PIP" "PPC" "PR CC" "PROG"
"RAMTOP" "RASP" "REPOEL"
8060 DATA "REPPER" "S POSN" "S T
OP" "SCR CT" "SEED" "SPOSNL" "ST
K BOT" "STKEND" "STLEN" "STRMS"
8070 DATA "SUBPPC" "T ADDR" "TVD

```

```

ATX" "TVFLAG" "UDG" "VAR5" "WORK
SP" "X PTR
8500 REM DIRECCIONES Y TIPOS
8510 DATA "123693" "123695" "123
624" "123655" "223645" "223631
" "223606" "423677" "223633" "2236
39
8520 DATA "223563" "223629" "223
634" "123659" "223686" "223641"
"223625" "223682" "123610" "2236
13
8530 DATA "123611" "123658" "123
655" "323672" "223643" "123585"
"523552" "123560" "223515" "1236
94
8540 DATA "123696" "223656" "223
698" "123617" "223618" "223728"
"123620" "223637" "223662" "1236
64
8550 DATA "123697" "123679" "223
733" "123609" "223621" "123880"
"223635" "223730" "123608" "1236
51
8560 DATA "123562" "423688" "223
680" "123692" "223670" "423690"
"223651" "223653" "223666" "6236
68
8570 DATA "123623" "223668" "223
586" "123612" "223675" "223627"
"223649" "223647"

```

Bytes		dirección		Contenido
Nombre		dec	hexa	
ATTR P	1	23693	5C8D	Atributos permanentes en curso
ATTR T	1	23695	5C8F	Atributos temporales en curso
BOROCR	1	23624	5C48	Atributos de la parte inferior de la pantalla
BREG	1	23655	5C67	Registro "B" del calculador
CH ADD .	2	23646	5C5D	Dirección del siguiente carácter que ha de ser interpretado por el intérprete de Basic.
CHANS	2	23631	5C4F	Dirección del área de información para canales
CHARS	2	23606	5C36	Dirección del juego de caracteres, menos 256.
COORDS	2	23677	5C7D	El primer byte indica la coordenada "x" del último punto PLOTado, y el segundo byte, la coordenada "y"
CURCHL	2	23633	5C51	Dirección del canal en curso
DATADD	2	23639	5C57	Puntero de DATAs.
DEFADD	2	23563	5C0B	Dirección del argumento de una función definida por el usuario si se está validando alguna, en otro caso, vale 0.
DEST	2	23629	5C4D	Dirección de la variable en una asignación
DF CC	2	23684	5C84	Dirección de la posición de PRINT en el archivo de pantalla
DF SZ	1	23659	5C68	Número de líneas, incluida una en blanco, de la parte inferior de la pantalla
DFCCL	2	23686	5C86	Como "DF CC", pero para la parte inferior de la pantalla.
E LINE	2	23641	5C59	Dirección del comando que está siendo tecleado.
E PPC	2	23625	5C49	Número de la línea a la que apunta el cursor del programa ">"
ECHO E	2	23682	5C82	Número de las 33 columnas, y de las 24 líneas del final del buffer de entrada
ERR NR	1	23610	5C3A	Código del informe, menos 1.
ERR SP	2	23613	5C3D	Dirección del elemento de la pila de máquina que se usa como retorno en caso de error
FLAGS	1	23611	5C3B	Indicadores de control del Sistema
FLAGS, 2	1	23658	5C6A	Cursor "L" o "C". "L" = 0, "C" = 8
FLAGX	1	23665	5C71	Indicadores de control del Sistema
FRAMES	3	23672	5C78	Reloj en tiempo real se incrementa cada 50 milisegundos.
K CUR	2	23643	5C5B	Dirección del cursor

Bytes		dirección		Contenido
Nombre		dec.	hexa.	
K DATA	1	23565	5C0D	Segundo byte de los controles de color introducidos por el teclado
KSTATE	8	23562	5C0E	Ocho variables intermedias, usadas para leer al teclado
LAST K	1	2356D	5C08	Código de la última tecla pulsada.
LJSTSP	2	23615	5C3F	Dirección de retorno tras un listado automático
MASK P	1	23694	5C8E	Máscara para colores transparentes
MASK T	1	23696	5C90	Como "MASK P", pero temporal
MEM	2	23666	5C68	Dirección del área usada como memoria del calculador
MEMBOT	2	23698	5C92	Base de la memoria del calculador
MODE	1	23617	5C41	Modo de cursor "K"=0, "E"=1, "G"=2
NEWPPC	2	23618	5C42	Número de línea a la que hay que saltar
NMI	2	23728	5CB0	Vector de interrupción no enmascarable (anulada para facilitar la protección del software comercial)
NSPPC	1	23620	5C44	Número de sentencia dentro de una línea, a la que hay que saltar
NXTLN	2	23637	5C56	Dirección de la siguiente línea de programa
OLOPPC	2	23662	5C6E	Línea a la que salta CONTINUE
OSPPC	1	23664	5C70	Número de sentencia, dentro de la línea a la que salta CONTINUE
P FLAG	1	23697	5C91	Indicadores para la impresión.
P POSN	1	23679	5C7F	Número de las 33 columnas de la posición de la impresora.
P RAMT	2	23732	5CB4	Dirección del último byte de RAM física.
PIP	1	23609	5C39	Duración del tono emitido al pulsar una tecla.
PPC	2	23621	5C45	Contador del programa en Basic (número de línea)
PR CC	1	23680	5CB0	Byte menos significativo de la dirección de la siguiente posición de LPRINT
PROG	2	23635	5C53	Dirección de inicio del programa Basic.
RAMTOP	2	23730	5CB2	Dirección del último byte del área de memoria ocupada por el sistema Basic
RASP	1	23608	5C38	Duración de la señal emitida al llenarse la memoria
REPDEL	1	23561	5C09	Tiempo (en 1/50 de segundo) que ha de estar pulsada una tecla para que comience a repetirse.
REPPER	1	23562	5C0A	Tiempo (en 1/50 de segundo) entre sucesivas repeticiones de una tecla
S POSN	2	23688	5CB8	El primer byte contiene el número de 33 columnas de la posición de PRINT, el segundo contiene el número de 24 líneas.
S TOP	2	23660	5C6C	Número de la línea superior en un listado automático.
SCR CT	1	23692	5C8C	Contador de (Scroll): es siempre 1 más que el número de líneas que se han de subir antes de preguntar (scroll?).
SEED	2	23670	5C76	Origen de las operaciones para generar un número pseudo-aleatorio
SPOSNL	2	23690	5C8A	Como (S POSN), pero para la parte inferior de la pantalla
STKBOT	2	23651	5C63	Dirección del fondo de la pila del calculador
STKEND	2	23653	5C65	Dirección de la parte superior de la pila del calculador e inicio del área de reserva
STRLEN	2	23666	5C72	Longitud de la cadena de destino en una asignación
STRMS	38	23568	5C10	Canales unidos a las corrientes abiertas
SUBPPC	1	23623	5C47	Contador de programa Basic (número de sentencia dentro de la línea)
T ADDR	2	23668	5C74	Dirección del siguiente elemento de la tabla sintáctica.
TVDATA	2	23566	5C0E	Bytes de color y controles AT y TAB que van al televisor
TVFLAG	1	23612	5C3C	Indicadores asociados con el televisor
UDG	2	23675	5C7B	Dirección del área de gráficos definidos por el usuario
VAR5	2	23627	5C4B	Dirección del área de variables.
WORKSP	2	23649	5C61	Dirección del área de trabajo.
X PTR	2	23647	5C5F	Dirección del carácter que sigue al signo (?)

EL JUEGO DE CARACTERES

Cada uno de los signos, letras o números que puede imprimir el Spectrum se corresponde con un número comprendido entre 0 y 255, a este número se le denomina *código*.

Existen varios sistemas normalizados de codificación de caracteres, el más utilizado en informática se denomina ASCII, y es el que utiliza el Spectrum, con ligeras variaciones.

EL ASCII

La palabra ASCII está compuesta por las siglas de «American Standard Code for Information Interchange» (Código Normalizado Americano para Intercambio de Información).

El ASCII se elaboró en los Estados Unidos a finales de la década de los 60. En principio se pensó para su uso en teletipos, estos aparatos transmiten 7 bits de código más uno de «paridad», por lo que el ASCII utiliza códigos comprendidos entre 0 y 127. La decodificación en los teletipos se hacía de forma mecánica, por lo cual este proceso tuvo que ser simplificado al máximo. El ASCII utiliza los cinco bits de menor peso para designar un carácter determinado, y los dos siguientes, para indicar si se trata de

	001.....	010.....	011.....
...00000	espacio	@	£
...00001	!	A	a
...00010	"	B	b
...00011	#	C	c
...00100	\$	D	d
...00101	%	E	e
...00110	&	F	f
...00111	'	G	g
...01000	(H	h
...01001)	I	i
...01010	*	J	j
...01011	+	K	k
...01100	,	L	l
...01101	-	M	m
...01110	.	N	n
...01111	/	O	o
...10000	0	P	p
...10001	1	Q	q
...10010	2	R	r
...10011	3	S	s
...10100	4	T	t
...10101	5	U	u
...10110	6	V	v
...10111	7	W	w
...11000	8	X	x
...11001	9	Y	y
...11010	:	Z	z
...11011	;	[{
...11100	<	\	

un código no imprimible (de control), un número o signo, una mayúscula o una minúscula.

Los 32 primeros códigos eran de control, y no producían la impresión de ningún carácter. Los 96 restantes constituyen propiamente, el juego de caracteres del ASCII, el código 127 se usa con frecuencia, para indicar el borrado del último carácter impreso. En la FIGURA 1 se pueden ver todos los caracteres del ASCII, ordenados según la configuración binaria de su código.

El juego de caracteres del Spectrum

El Spectrum utiliza una variante del ASCII, los 32 primeros códigos son también de control, si bien cumplen funciones bastante diferentes a las asignadas por el ASCII. Los 96 siguientes son idénticos a los caracteres ASCII, salvo el código 127 que se utiliza para el signo de «Copyright».

Por otro lado, el Spectrum no necesita comprobación de paridad, por lo que los ocho bits están disponibles. Ello permite duplicar el número de códigos utilizables.

Los 37 códigos siguientes al 127, se han utilizado para caracteres gráficos, de los cuales, los primeros 16 están definidos, y los 21 restantes son definibles por el usuario (los famosos UDG).

Los últimos 91 códigos han sido asignados a los «Tokens» que utiliza el Spectrum. La función CODE aplicada sobre una cadena, da como resultado el código del primer carácter que la compone. Su

...11101	=	1)
...11110	>	^	~
...11111	?	-	█

Figura 1. Configuración binaria del ASCII.

Código		Función
Dec.	Hexa	
0	00	No Utilizado
1	01	No utilizado
2	02	No utilizado
3	03	No utilizado
4	04	No utilizado
5	05	No utilizado
6	06	CAPS LOCK y efecto de "coma" en impresión.
7	07	EDIT
8	08	Cursor Izquierda
9	09	Cursor Derecha
10	0A	Cursor Abajo
11	0B	Cursor Arriba
12	0C	DELETE (Borrado)
13	0D	ENTER (Nueva línea)
14	0E	CAPS SHIFT + SYMBOL SHIFT y "número" dentro de un programa.
15	0F	No Utilizado
16	10	INK (Control de tinta).
17	11	PAPER (Control de papel).
18	12	FLASH (Control de parpadeo).
19	13	BRIGHT (Control de brillo)
20	14	INVERSE (Control de inversión de video).
21	15	OVER (Control de sobrepresión).
22	16	AT (Control de posicionamiento).
23	17	TAB (Control de tabulación)
24	18	No Utilizado
25	19	No Utilizado
26	1A	No Utilizado
27	1B	No Utilizado
28	1C	No Utilizado
29	1D	No Utilizado
30	1E	No Utilizado
31	1F	No Utilizado

Figura 2. Códigos de Control.

inversa, CHR\$ aplicada sobre un número, da como resultado el carácter que tiene ese número como código.

La FIGURA 2, muestra la lista de códigos de control utilizados en el Spectrum. Las FIGURAS 3, 4 y 5 muestran las

tres partes de juego de caracteres. La FIGURA 6 corresponde a los caracteres gráficos y por último, la FIGURA 7 muestra la lista de «Tokens»

con sus códigos.

En todos los casos, los códigos han sido representados en decimal como en hexa. ■

Código		Caracter
Dec.	Hexa	
32	20	Espacio
33	21	,
34	22	.
35	23	:
36	24	;
37	25	'
38	26	"
39	27	'
40	28	!
41	29)
42	2A	@
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Figura 3. Signos y números.

Codigo		Caracter
Dec.	Hexa	
64	40	Q
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D]
94	5E	↑
95	5F	-

Figura 4. Letras mayúsculas.

Codigo		Caracter
Dec.	Hexa	
96	60	£
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	
125	7D	}
126	7E	*
127	7F	(c)

Figura 5. Letras minúsculas.

Codigo		Caracter
Dec.	Hexa	
128	80	
129	81	.
130	82	:
131	83	-
132	84	;
133	85	!
134	86	%
135	87	^
136	88	'
137	89	/
138	8A]
139	8B	~
140	8C	=
141	8D	^
142	8E	~
143	8F	~
144	90	UDG "A"
145	91	UDG "B"
146	92	UDG "C"
147	93	UDG "D"
148	94	UDG "E"
149	95	UDG "F"
150	96	UDG "G"
151	97	UDG "H"
152	98	UDG "I"
153	99	UDG "J"
154	9A	UDG "K"
155	9B	UDG "L"
156	9C	UDG "M"
157	9D	UDG "N"
158	9E	UDG "O"
159	9F	UDG "P"
160	A0	UDG "Q"
161	A1	UDG "R"
162	A2	UDG "S"
163	A3	UDG "T"
164	A4	UDG "U"

Figura 6. Caracteres gráficos.

Codigo		Token
Dec.	Hexa	
165	A5	RND
166	A6	INKEY\$
167	A7	PI
168	A8	FN
169	A9	POINT
170	AA	SCREEN\$
171	AB	ATTR
172	AC	AT
173	AD	TAB
174	AE	VAL\$
175	AF	CODE
176	B0	VAL
177	B1	LEN
178	B2	SIN
179	B3	COS
180	B4	TAN
181	B5	ASN
182	B6	ACS
183	B7	ATN
184	B8	LN
185	B9	EXP
186	BA	INT
187	BB	SQR
188	BC	SGN
189	BD	ABS
190	BE	PEEK
191	BF	IN
192	C0	USR
193	C1	STR\$
194	C2	CHR\$
195	C3	NOT
196	C4	BIN
197	C5	OR
198	C6	AND
199	C7	<=
200	C8	>=
201	C9	<>
202	CA	LINE
203	CB	THEN
204	CC	TO
205	CD	STEP
206	CE	DEF FN
207	CF	CAT
208	D0	FORMAT
209	D1	MOVE
210	D2	ERASE

Figura 7.

Códigos de «Tokens».

Codigo		Token
Dec.	Hexa	
211	D3	OPEN #
212	D4	CLOSE #
213	D5	MERGE
214	D6	VERIFY
215	D7	BEEP
216	D8	CIRCLE
217	D9	INK
218	DA	PAPER
219	DB	FLASH
220	DC	BRIGHT
221	DD	INVERSE
222	DE	OVER
223	DF	OUT
224	E0	LPRINT
225	E1	LLIST
226	E2	STOP
227	E3	READ
228	E4	DATA
229	E5	RESTORE
230	E6	NEW
231	E7	BORDER
232	E8	CON: PUE
233	E9	DIM
234	EA	REM
235	EB	FOR
236	EC	GO TO
237	ED	GO SUB
238	EE	INPUT
239	EF	LOAD
240	F0	LIST
241	F1	LET
242	F2	PAUSE
243	F3	NEXT
244	F4	POKE
245	F5	PRINT
246	F6	PLOT
247	F7	RUN
248	F8	SAVE
249	F9	RANDOMIZE
250	FA	IF
251	FB	CLS
252	FC	DRAW
253	FD	CLEAR
254	FE	RETURN
255	FF	COPY

INDICE

INTRODUCCION

El lenguaje del Spectrum _____

Parte I

Capítulo 1

TECLADO DEL SPECTRUM

Acceso al teclado _____

Modo **[K]** _____

Modo **[L]** _____

Modo **[C]** _____

Modo **[E]** _____

Modo **[G]** _____

Capítulo 2

CONFECCION DE PROGRAMAS

Edición de programas _____

Corrección de errores _____

Ejercicio _____

Capítulo 3

EL SPECTRUM PLUS

Teclado del «ZX Spectrum +» _____

Modos **[L]** **[C]** _____

Modo **[E]** _____ 15

Modo **[G]** _____ 16

2 Edición de programas _____ 16

Capítulo 4

ALMACENAMIENTO DE PROGRAMAS

Introducción _____ 17

Verificación _____ 19

3 Recuperación de programas _____ 20

6 Protección de programas _____ 21

6 Conservación de cintas _____ 23

6 Oscilación _____ 24

7 Ajuste _____ 24

Capítulo 5

CONSTANTES Y VARIABLES

Introducción _____ 25

Constantes numéricas _____ 26

11 Notación entera _____ 26

14 Notación decimal _____ 27

Notación exponencial _____ 27

Notación binaria _____ 30

Decimal Binario _____ 30

Binario-Decimal _____ 31

Ejercicio _____ 31

Constantes alfanuméricas _____ 31

14 Variables numéricas _____ 32

15 Variables alfanuméricas _____ 32

Capítulo 6

OPERADORES

Introducción	33
Operadores aritméticos	33
Expresiones aritméticas	33
Cálculo de expresiones	33
Operadores de relación	34
Operadores lógicos	34
Función «AND»	34
Función «OR»	35
Función «NOT»	35
Ejercicio	36

Capítulo 7

CODIGO ASCII

Introducción	37
Manejo de la tabla	37
Organización del ASCII	38
Transmisión del ASCII	40

Capítulo 8

OPERACIONES CON CADENAS

Concatenación de cadenas	42
Subcadenas	43
Fragmentación	43
Fragmentación específica	43
Asignación de subcadenas	45
Comparación de cadenas	45
Ordenación de cadenas	47
Prioridades	48

Capítulo 9

ELABORACION DE PROGRAMAS

320 MICROBASIC

Introducción	50
Análisis	50
Síntesis	51
Representación gráfica	51
Programa	52

Parte II

Capítulo 10

EL JUEGO DE SENTENCIAS

Clasificación	60
Comandos de control	60
Comandos de programación	60
Comandos de entrada/salida	60
Manejo de cadenas	61
Funciones aritméticas	61
Funciones lógicas	61
Comandos de dibujo	61
Comandos de control de color	62
Sonido	62
Manejo de impresora	62
Interface 1	63
Manejo Microdrive	63
Auxiliares	64
Programa «COLOREAR»	64
Programa «GRAFICAS»	64
Programa «GDU»	66
Programa «DIBUJANDO»	66
Programa «BIPBIP»	68

Capítulo 11

COMANDOS BASICOS

REM	69
Funciones de Video	70
IF	70
Representación gráfica	71
PRINT	71
Desplazamientos	72
Formatos	72

TAB	73
AT	73
Canales de comunicación	75
INPUT	77
INPUT TAB y AT	79
INPUT LINE	80
Otra aplicación	80
Programas de repaso	81
Programa «GRANJA»	81
Programa «EDUCACION»	83
Programa «INTERES»	84
Programa «GRADOS»	86
Programa «FICHA»	86

Capítulo 12

COMANDOS DE CONTROL

RUN	89
BREAK	90
STOP	91
Ruptura del «INPUT»	92
CONTINUE	94
Informes de pantalla	96
NEW	97
CLS	97
LIST	98
LIST y EDIT	98

Capítulo 13

SALTOS INCONDICIONALES Y CONDICIONALES

Introducción	99
GO TO	99
IF... THEN...	101
Evaluación de las condiciones	103
Programa	107

Capítulo 14

BUCLES

Introducción	108
DO WHILE	108
REPEAT UNTIL	108
Diferencias	108
FOR/NEXT	109
STEP	111
Bucles anidados	112
Errores	113
Programas	113

Capítulo 15

SUBROUTINAS

Introducción	122
GO SUB	122
RETURN	122
Utilización de «GOSUB» y «RETURN»	122
Tipos de subrutinas	124
Subrutinas anidadas	125
Error	126
Programas	127

Capítulo 16

DATOS DE UN PROGRAMA

Introducción	134
READ	134
DATA	134
Utilización de «READ» y «DATA»	134
RESTORE	135
Errores	137
Programas	140

Capítulo 17

LECTURA DEL TECLADO Y TEMPORIZACIONES

INKEY\$	142
PAUSE	142
Programas	144

Capítulo 18

FUNCIONES

Introducción	148
Funciones numéricas	148
ABS	148
INT	148
SGN	149
SQR	149
DIN	150
PI	150
El radián	151
SIN	151
COS	152
TAN	152
ASN	152
ACS	152
ATN	152
Aplicación de la trigonometría	153
Función exponencial	154
EXP	154
Función logarítmica	155
LN	155
Definición de funciones	159
DEF FN	159
FN	159
Errores	160

Capítulo 19

FUNCION ALEATORIA

Introducción	161
RND	161
Programa «BARQUITOS»	164
RANDOMIZE	164
Programa «TABLA»	168

Capítulo 20

FUNCIONES DE CADENA

Introducción	169
--------------	-----

LEN	170
STR\$	171
VAL	171
VAL\$	172
Conversiones de código	172
CHR\$	172
CODE	174
Funciones definidas de cadena	175
Errores	175
Programa	177

Capítulo 21

MATRICES

Introducción	179
Dimensionado de matrices	179
DIM	180
Matrices numéricas	180
Asignación y visualización	182
Manejo de tablas	185
Matrices de cadena	195
Asignación	196
Fragmentación	196
Errores	199
Grabación de datos	200
Programa	202

Capítulo 22

DEPURACION DE PROGRAMAS

Introducción	206
Errores	206
Depuración	207
STOP y CONTINUE	209
Programa «Depurador»	209
Ejercicio	210

Capítulo 23

COLOR

Introducción	211
--------------	-----

Teoría del color	212
Síntesis aditiva	213
Zonas de pantalla	214
BORDER	214
PAPER	215
INK	215
Atributos permanentes y temporales	216
Resolución del color	217
Transparencia y contraste	217
Simulación de colores	218
Control de impresión	219
BRIGHT	219
INVERSE	220
FLASH	221
Atributos de pantalla	222
ATTR	222
Caracteres de control	222
Acceso directo	223
Errores	224

Capítulo 24

GRAFICOS

Introducción	226
Tipos de gráficos	226
Bloques de color	226
Gráficos predefinidos	226
Pantalla en alta resolución	228
PLOT	228
DRAW	229
Arcos de circunferencia	232
Programa especial	234
CIRCLE	234
Técnicas avanzadas	235
OVER	239
SCREEN\$	242
Almacenamiento de pantallas	242
POINT	242
Programa	243

Capítulo 25

GRAFICOS DEFINIDOS

Introducción	246
¿Cómo se almacenan?	246
Definición de «GDU»	249
Utilización de los «GDU»	250
Programas de aplicación	254
Grabación de los «GDU»	254
Lectura de los «GDU»	254
Programa generador de «GDU»	254
Programa	255

Capítulo 26

SONIDO

Introducción	258
BEEP	258
Nociones musicales	261
Tono	261
Duración	263
Compás	264
Variables relacionadas	264
Grabación de sonidos	264
Periféricos	266
Software Musical	266
Efectos sonoros	266

Capítulo 27

SENTENCIAS DE GRABACION Y CARGA

Introducción	267
SAVE	267
Programa LISTADOR	268
Grabación de matrices	269
Programa DIRECTORIO	270
VERIFY	271
LOAD	272
Programa EDIT/DIR	273
MERGE	273
Comodidad de uso	274
Búsqueda de programas	274
Referencias	274
Programas	274

Capítulo 28

GESTION DE IMPRESORA

Introducción	276
LPRINT	276
Programa TEST DE IMPRESORA	277
LLIST	277
COPY	278
Ejemplo de COPY	278
Otras impresoras	278
Tipos de impresoras	279
Elección de una impresora	280
Juego de caracteres	281

Capítulo 29

INTERFACE 1

Introducción	283
Canales y corrientes	283
Asignación de canales y corrientes	283
OPEN #	283
Desactivación de canales y corrientes	284
CLOSE #	284
El Microdrive	284
FORMAT	284
Programa FICHERO	285
CAT	286
Grabación y carga	287
Borrado de programas	287
ERASE	288
Ejecución automática	288
Protección de ficheros	288
Ficheros de datos	288
Grabación de datos	289
Apertura de fichero	289
Cierre de ficheros	289
Lectura y ampliación de ficheros	290
Red de área local	292
Interface RS-232	293

Capítulo 30

LA MEMORIA

Introducción	295
POKE	295
PEEK	296
Tipos de memoria	296
Bit y Byte	297
Memoria ROM	297
La Memoria RAM	298
Almacenamiento de programas	301
Variables	301
Variable numerica cuyo nombre es una sola letra	301
Variable numerica cuyo nombre son varias letras	301
Variable de cadena de caracteres	301
Variable de control de bucle FOR-NEXT	301
Matriz de numeros	302
Matriz de caracteres	302
Borrado de variables	302
CLEAR	302
Cuando se llena la memoria	303
Programando en codigo maquina	305
USR	305

Capítulo 31

LOS PERIFERICOS

Introducción	307
OUT	308
IN	308
Ports del teclado	309
Programa SCAN DEL TECLADO	311
Programa SKETCH	311

Capítulo 32

VARIABLES DEL SISTEMA

Introducción	312
Programa LECTOR DE VARIABLES	312
Tabla de variables del Sistema	313

SENTENCIAS Y FUNCIONES

A

ABS	148
ACS	152
AND	34
ASN	152
AT	73
ATN	152
ATTR	222

B

BEEP	258
BIN	150
BORDER	214
BRIGHT	219

C

CAT	286
CHR\$	172
CIRCLE	234
CLEAR	302
CLOSE #	284
CLS	97
CODE	174
CONTINUE (CONT)	94 y 209
COPY	278
COS	152

D

DATA	134
DEF FN	159
DIM	180
DRAW	229

E

ERASE	288
EXP	154

F

FLASH	221
FN	159
FOR	109
FORMAT	284

G

GOSUB	122
GO TO	99

I

IF	101
IN	308
INK	215
INKEY\$	142
INPUT	77
INT	148
INVERSE	220

L

LEN	170
LET	70
LINE	80
LIST	98
LLIST	277
LN	155
LOAD	272

LPRINT _____ 276

M

MERGE _____ 273

MOVE _____ 292

N

NEW _____ 97

NEST _____ 109

NOT _____ 35

O

OPEN # _____ 283

OR _____ 35

OUT _____ 308

OVER _____ 239

P

PAPER _____ 215

PAUSE _____ 142

PEEK _____ 296

PI _____ 150

PLOT _____ 228

POINT _____ 242

POKE _____ 295

PRINT _____ 71

R

RANDOMIZE (RAND) _____ 164

READ _____ 134

REM _____ 69

RESTORE _____ 135

RETURN _____ 122

RND _____ 161

RUN _____ 89

S

SAVE _____ 267

SCREEN\$ _____ 242

SGN _____ 149

SIN _____ 151

SQR _____ 149

STEP _____ 111

STOP _____ 91 y 209

STR\$ _____ 171

T

TAB _____ 73

TAN _____ 152

THEN _____ 101

TO _____ 110

U

USR _____ 305

V

VAL _____ 171

VAL\$ _____ 172

VERIFY _____ 271

PROGRAMAS «MICROBASIC»

EJER1	10, 11, 15 y 16	CARTAS	189, 190 y 191
CODEBIN	25, 26 y 27	11 ERRORES	209
ASCII	37	SIN ERRORES	210
ASCII/DECIMAL	39	ATRIBUTOS	224
CHR\$	40	DEPURACION	225
CONCATENACION	42	CARTA COLOR	225
FRAGMENTACION	44 y 45	COLORES 1	225
COMPARACION	47	COLORES 2	225 y 226
ORDENA	48 y 49	DIBUJO PEZ	227 y 228
LISTIN	51, 52, 53 y 59	MICROHOBBY	230
COLOREAR	61	ABSTRACTO	232
GRAFICAS	62	GRAFICO	233
G.D.U.	64	RECTAS	234
DIBUJANDO	65	MAPA	237, 238 y 239
BIPBIP	66	OVER	239
GRANJA	74 y 75	LABERINTO	240
EDUCACION 2 GRADO	75 y 76	ALFABETO ESPAÑOL	246
INTERES SIMPLE	77	NOTAS GRAFICAS	247
GRADOS	78 y 79	LECTURA «GDU»	250
FICHA	79 y 80	PALITROQUE	251, 252 y 253
CALCULADORA	92 y 93	NAVIDAD	259
iiiNEW!!!	93 y 94	DIATONICA	259
AGENDA	102	CROMATICA	260
AREAS Y PERIMETROS	105 y 106	DOS CRUCES	260 y 261
ESTADISTICA	114 y 115	EFECTO 1	263
HISTOGRAMAS	116	EFECTO 2	263
ADIVINO	126	EFECTO 3	263
LONGITUD	130	EFECTO 4	263
GEOGRAFIA 1	138	NAVIDAD (TREMOLO)	264
GEOGRAFIA 2	139 y 140	LISTADOR	268
MAQUINA	145	DIRECTORIO	270 y 271
MOVIMIENTO	157	EDIT/DIR	273
iiiAGUA!!!	165	IMPRESORA (TEST)	277
LA TABLA	166	FICHERO	285 y 286
BUSQUEDA	169	RENUMERADOR	304 y 306
INSERTAR	169	SCAN DE TECLADO	311
ANULAR	171	SKETCK	311
INPUT	174	LECTURA DE VARIABLES	312
MANEJO DE TABLAS	179 y 180		

MICROBASIC

Aunque el Basic fue diseñado como un lenguaje de alto nivel universal para todos los microordenadores, la salida al mercado de los distintos modelos fue diversificando su estructura y prestaciones hasta tal punto que actualmente puede decirse que no hay dos máquinas que permitan ser programadas utilizando un Basic idéntico para ambas.

Sinclair no constituye una excepción y este manual está especializado en el Basic utilizado en sus microordenadores Spectrum y Spectrum+.

Profusamente ilustrado con esquemas, fotos y figuras a todo color, todas las explicaciones están acompañadas de abundantes ejemplos prácticos.

Con toda probabilidad, se trata del libro más completo que se ha publicado sobre el Basic Sinclair, aclarando importantes lagunas existentes en este terreno.

Respecto de la programación en Basic, constituye una completa guía de utilización de cada uno de los comandos disponibles y, quizás el aspecto más destacable, su grado de dificultad es progresivo, haciéndolo especialmente indicado para todas aquellas personas que sin tener conocimientos específicos en esta materia desean aventurarse en el apasionante mundo de la programación.